Conf.univ.dr. Ion Lungu
Conf.univ.dr. Ion Roșca
Economist Nely Mușat
Conf.univ.dr. Gh. Sabău

BAZE DE DATE RELAȚIONALE

UTILIZAREA LIMBAJULUI

SQL*PLUS

Biblioteca de Informatică

Editura ALL

BAZE DE DATE RELAȚIONALE

UTILIZAREA LIMBAJULUI SQL*PLUS

Apavăloaie
Dumitru
Daniel

Scanned with OKEN Scanner

BAZE DE DATE RELAȚIONALE
UTILIZAREA LIMBAJULUI SQL*PLUS

Conf. univ. *dr.* **Ion Lungu**
Economist **Nely Mușat**
Conf. univ. *dr.* **Ion Roșca**
Conf. univ. *dr.* **Gh. Sabău**

Editura ALL București 1992

Scanned with OKEN Scanner

CUPRINS

INTRODUCERE 1

CAPITOLUL 1
Concepte utilizate şi elementele de bază ale limbajului SQL*PLUS
1.1. Concepte utilizate 3
1.2. Nume şi cuvinte rezervate 6
1.3. Operatorii şi funcţiile utilizate în comenzile SQL 6

CAPITOLUL 2
Conectarea şi deconectarea unei baze de date
2.1. Deschiderea unei sesiuni de lucru SQL*PLUS 7
2.2. Conectarea utilizatorului la o bază de date 8
2.3. Deconectarea utilizatorului de la baza de date 9
2.4. Închiderea sesiunii de lucru SQL*PLUS 10

CAPITOLUL 3
Editarea unui program SQL
3.1. Introducerea liniilor de comandă 11
3.2. Adăugarea unui text 12
3.3. Înlocuirea unui text vechi cu unul nou 13
3.4. Ştergerea unei linii curente 14
3.5. Listarea liniilor din buffer-ul curent 14
3.6. Salvarea programului din buffer-ul curent 15
3.7. Încărcarea conţinutului unui fişier de tip SQL în buffer-ul curent 16
3.8. Lansarea în execuţie a unui alt editor 16

Scanned with OKEN Scanner

CAPITOLUL 4
Selectarea datelor dintr-o bază de date
4.1. Comanda SELECT 17
4.2. Utilizarea clauzei FROM 19
4.3. Utilizarea operatorilor în formularea condiţiilor de selecţie din clauza WHERE 21
4.4. Ordonarea liniilor rezultate în urma unei cereri 31
4.5. Selecţii din mai multe tabele 33
4.6. Realizarea cererilor incluse 38
4.7. Utilizarea expresiilor, funcţiilor, variabi-lelor sistem şi pseudo-coloanelor în selectarea datelor 45
4.8. Utilizarea funcţiilor de grupare şi a clauzei GROUP BY în selectarea datelor 63
4.9. Operaţii pe tabele structurate arborescent 72

CAPITOLUL 5
Actualizarea datelor
5.1. Adăugarea unor rînduri noi 83
5.2. Ştergerea rîndurilor dintr-o tabelă 84
5.3. Modificarea datelor dintr-o tabelă 86

CAPITOLUL 6
Definirea unei baze de date
6.1. Crearea şi modificarea unei partiţii 88
6.2. Definirea, modificarea şi ştergerea unui model de alocare a spaţiului 89
6.3. Crearea, modificarea şi ştergerea unui cluster 92
6.4. Crearea, modificarea şi ştergerea unei tabele 96
6.5. Crearea şi ştergerea sinonimelor pentru tabele sau viziuni 102
6.6. Crearea şi ştergerea unei viziuni 103
6.7. Crearea, validarea şi ştergerea indecşilor pentru o tabelă 105

Scanned with OKEN Scanner

CAPITOLUL 7
Controlul accesului la baza de date
7.1. Acordarea drepturilor de acces 109
7.2. Revocarea drepturilor de acces 111
7.3. Reţinerea în baza de date a comenzilor utilizator 111
7.4. Renunţarea la reţinerea în bază a comenzilor utilizator 113
7.5. Partajarea accesului la tabele 114

CAPITOLUL 8
Definirea opţiunilor de lucru şi realizarea altor operaţii
8.1. Setarea opţiunilor de lucru 116
8.2. Afişarea valorilor variabilelor de sistem 124
8.3. Evaluarea timpului necesar executării comenzilor 125
8.4. Stabilirea rîndurilor avansate pe următoarea pagină 126
8.5. Lansarea în execuţie a comenzilor SQL 128
8.6. Utilizarea variabilelor utilizator 130
8.7. Introducerea de comentarii printre comenzi 133
8.8. Introducerea de pauze printre comenzile SQL 134
8.9. Afişarea unor informaţii 134
8.10. Memorarea sesiunii de lucru într-un fişier 136
8.11. Lansarea de comenzi corespunzătoare sistemului de operare 136
8.12. Permanantizarea sau renunţarea la modificările făcute asupra bazei de date 137
8.13. Copierea datelor între baze de date diferite 138

CAPITOLUL 9
Formatarea situaţiilor de ieşire
9.1. Tratarea întreruperilor 140
9.2. Tipărirea titlurilor 143
9.3. Anularea opţiunilor 149
9.4. Descrierea coloanelor 149
9.5. Realizarea de calcule 155

Scanned with OKEN Scanner

CAPITOLUL 10
Comenzi pentru lucrul în rețea
10.1. Crearea legăturii cu baza de date de la distanță 162
10.2. Ștergerea legăturii cu baza de date de la distanță 163

ANEXA 1
CUVINTE REZERVATE 164

ANEXA 2
*OPERATORI UTILIZAȚI ÎN LIMBAJUL SQL*PLUS* 166

ANEXA 3
FUNCȚII, VARIABILE SISTEM ȘI PSEUDO-COLOANE 169

ANEXA 4
DICȚIONARUL DE CUVINTE/EXPRESII 176

BIBLIOGRAFIE 180

Scanned with OKEN Scanner

INTRODUCERE

Sistemele de conducere a bazelor de date cu structuri arborescente şi reţea, datorită unor limite tehnice precum şi lipsei de flexibilitate, sînt înlocuite din ce în ce mai mult cu sistemele de conducere a bazelor de date relaţionale (Relational Database Management System-RDMS).

Bazele de date relaţionale, limbajele generaţiei a IV-a şi generatoarele de aplicaţii sînt concepte care au cîştigat mulţi adepţi din lumea informaticii. Autorii acestor concepte au pus la punct sisteme de gestiune a datelor şi de dezvoltare a aplicaţiilor din ce în ce mai puternice. Dintre aceste sisteme fac parte: *ORACLE, DBASE, FOXPRO, OMNIS 5, PARADOX, SQL SYSTEM, SQL BASE* etc.

Sistemul ORACLE este produs al firmei cu acelaşi nume. Ea a fost printre primele companii ce a oferit un sistem comercial pentru gestiunea bazelor de date relaţionale. Cercetările realizate pentru dezvoltarea componentelor acestui sistem au condus la apariţia versiunii a 6-a, cu multiple facilităţi faţă de versiunile anterioare.

Succesul obţinut de acest sistem se datorează multiplelor funcţiuni oferite. El are încorporat, începînd cu prima versiune, **limbajul SQL**. Este un sistem compatibil pe marea majoritate a calculatoarelor care se comercializează, de la microcalculatoare la calculatoare mari. Datele şi aplicaţiile create cu sistemul ORACLE sînt portabile pe calculatoare de modele şi dimensiuni diferite.

Sistemul ORACLE permite, de asemenea, accesul la fişierele bazei de date create cu alte sisteme sau limbaje de programare. Permite accesul interactiv la date, simplifică activitatea de programare şi asigură securitatea şi confidenţialitatea datelor, reduce spaţiul de memorare şi redundanţa datelor, asigură independenţa memorării datelor şi structura logică a lor.

Începînd cu *versiune 5.0.* acest sistem este implementat pe o structură de tip *"CLIENT-SERVER"*. Server-ul ORACLE include soft-ul de conducere a bazei de date şi cîteva instrumente necesare utilizatorilor şi administratorului bazei de date pentru actualizarea, afişarea şi utilizarea datelor. Dintre utilitarele incluse în Server-ul Oracle fac parte: *ORACLE Server Manager, SQL*DBA, Export / Import, SQL*Loader*. Sistemul ORACLE oferă, de asemenea, o serie de produse cum ar fi: *SQL*PLUS, SQL*FORMS, SQL*Report Writer, SQL*MENU, PRO*C, PRO*COBOL, PRO*FORTRAN, SQL*NET, ORACLE for dBASE interpreter, ORACLE Quicksilver, CASE*Method,*

Scanned with OKEN Scanner

*CASE*Dictionary etc.*

Componenta de bază a sistemului ORACLE este SQL. El este un limbaj de acces la date, utilizat pentru cele mai multe activităţi de gestiune a bazelor de date relaţionale. Este uşor accesibil utilizatorilor începători, dar, totodată, puternic, oferind programatorilor facilităţi deosebite. A luat naştere în laboratoarele de cercetare ale firmei IBM. Integrarea lui a fost făcută pentru prima dată în sistemul de gestiune a bazelor de date relaţionale "R". Numele lui este acronimul de la Structured Query Language (limbaj de interogare structurat). Datorită puterii sale relaţionale, uşurinţei însuşirii şi utilizării, a fost ales, în 1986, de ANSI şi apoi de ISO, ca limbaj standard pentru lucrul cu bazele de date relaţionale.

Cele mai multe firme au implementat limbajul SQL în sistemele de gestiune a bazelor de date realizate. Exemplul concludent îl reprezintă firma Ashton-Tate, care a lansat pe piaţă dBASE IV, ce are ca principală noutate faţă de dBASE III Plus, limbajul SQL. De asemenea, firma Borland a lansat pe piaţă *sistemul Paradox*, versiunea 3.0, care are componenta Paradox SQL.

Plecînd de la aspectele enunţate, această lucrare prezintă modul de utilizare a limbajului SQL*PLUS, versiunea 2.1 sub sistemul ORACLE 5.1.B. Prezentarea s-a făcut plecînd de la principalele grupe de operaţii ce pot fi realizate asupra datelor dintr-o bază de date relaţională şi punînd în evidenţă comenzile SQL şi SQL*PLUS, precum şi modul lor de utilizare.

Scanned with OKEN Scanner

CAPITOLUL 1
CONCEPTELE UTILIZATE ŞI ELEMENTELE DE BAZĂ ALE LIMBAJULUI SQL*PLUS

Limbajul SQL*PLUS este o dezvoltare a limbajului SQL. El cuprinde comenzi SQL şi comenzi SQL*PLUS. Un program scris în limbajul SQL*PLUS este format din una sau mai multe comenzi care la rîndul lor conţin clauze, nume de date, operatori, funcţii, pseudo-coloane etc. Ordinea scrierii în program a acestor comenzi şi a clauzelor din interiorul comenzilor este dată de funcţiile pe care trebuie să le realizeze programul şi de anumite restricţii de sintaxă.

1.1. Concepte utilizate

SQL*PLUS este un limbaj neprocedural şi operează asupra datelor normalizate. Conceptele necesare a fi cunoscute pentru lucrul cu acest limbaj sînt: *tabelă, cheie primară, coloană, rînd, viziune, index, sinonim, cluster, bază de date relaţională etc.*

Tabela sau ***relaţia*** este un ansamblu format din ***n*** coloane (atribute/subansambluri) şi ***m*** rînduri (tupluri/linii) care respectă următoarele condiţii minime: să nu existe date la nivel agregat (valorile aflate la intersecţia liniilor cu coloanele să fie la un nivel elementar); liniile să fie distincte unele faţă de altele; să nu existe coloane repetitive în descriere.Tabelele pot fi într-una din cele cinci forme normale: FN1, FN2, FN3, FN4 şi FN5.

O tabelă este în FN1 dacă toate atributele conţin valori elementare şi nu există coloane repetitive în descriere.

Un atribut B al unei tabele depinde funcţional de atributul A al aceleiaşi tabele dacă fiecărei valori a lui A îi corespunde numai o valoare a lui B. Un atribut B al unei tabele depinde funcţional complet de un ansamblu de atribute ale tabelei dacă atributui B este dependent funcţional de fiecare atribut din ansamblu.

O tabelă este în FN2 dacă este în FN1 şi fiecare atribut non-cheie este dependent funcţional de atributul cheie.

Fie atributele A,B şi C dintr-o tabelă. Dacă B depinde de A şi C

Scanned with OKEN Scanner

depinde de B, atunci C se află în dependenţă tranzitivă faţă de A.

O tabelă este în FN3 dacă este în FN2 şi fiecare atribut non-cheie depinde în mod netranzitiv de cîmpul cheie al tabelei.

Dacă valorii unui atribut A al unei tabele îi corespund două sau mai multe valori ale atributului B al aceleiaşi tabele, există o dependenţă multivaloare a lui B faţă de A.

O tabelă este în FN4 dacă este în FN3 şi nu conţine două sau mai multe dependenţe multivaloare.

O tabelă este în FN5 dacă este în FN4 şi fiecare joncţiune este generată printr-un atribut "candidat cheie" al tabelei.

Cheia primară este un atribut care are valori distincte. Deci, fiecare linie se identifică printr-o valoare distinctă. Două sau mai multe atribute care pot fi chei primare se numesc ***chei candidate***.

Coloana tabelei este formată din valorile pe care le ia atributul în liniile tabelei respective.

Rîndul/tuplul/linia este format din valorile coloanelor ce se referă la o entitate a tabelei.

Baza de date relaţională este un ansamblu de tabele normalizate, grupate în jurul unui subiect, în principiu, bine definit. Într-o bază de date relaţională, entităţile şi legăturile sînt transpuse în tabele.

Exemplele prezentate în capitolele ce urmează sînt realizate pe o bază de date relaţională privind activitatea comercială a unei firme. Schema conceptuală a acestei baze de date, în FN1, este prezentată în figura 1.1.

Viziunea este o tabela logică şi reprezintă o fereastră la date, dintr-una sau mai multe tabele.

Pentru ca accesul la date sa se facă mai rapid, se utilizează indexarea. Un ***index*** reprezintă o cheie pe una sau mai multe coloane. Indexarea este dinamică deoarece se pot adauga sau şterge indecşi oricînd, fără ca datele memorate sau aplicaţiile scrise să fie afectate.

Pentru realizarea unor operaţii sau pentru a utiliza în cereri nume mai scurte, se pot defini ***sinonime*** ale unor nume de tabele sau viziuni.

Un ***cluster*** reprezintă o anumită modalitate de grupare a rîndurilor uneia sau mai multor tabele. Această grupare măreşte viteza de execuţie a unor operaţii consumatoare de timp.

Scanned with OKEN Scanner

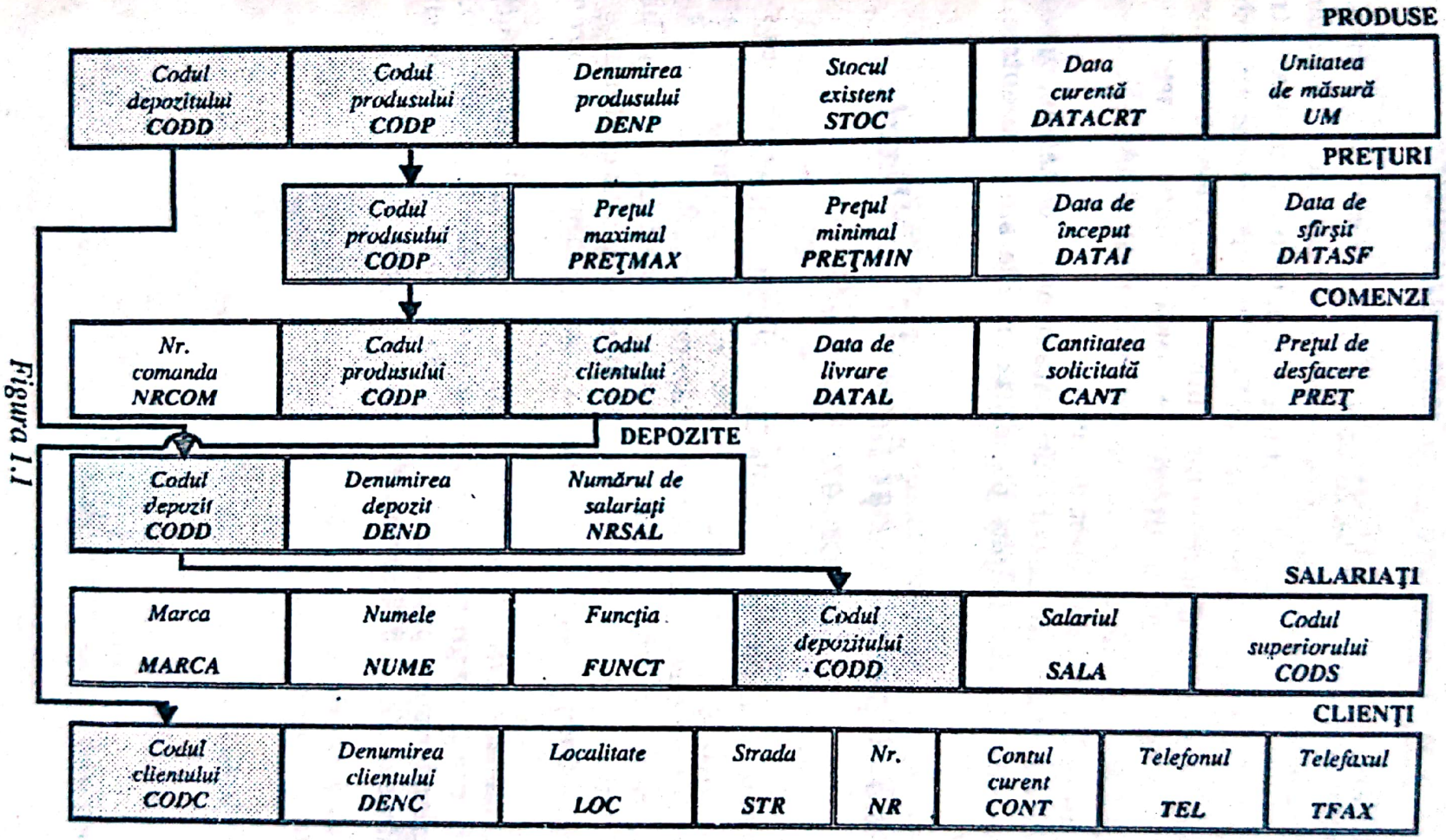
PRODUSE
Codul depozitului CODD
Codul produsului CODP
Denumirea produsului DENP
Stocul existent STOC
Data curentă DATACRT
Unitatea de măsură UM
PREŢURI
Codul produsului CODP
Preţul maximal PREŢMAX
Preţul minimal PREŢMIN
Data de început DATAI
Data de sfîrşit DATASF
COMENZI
Nr. comanda NRCOM
Codul produsului CODP
Codul clientului CODC
Data de livrare DATAL
Cantitatea solicitată CANT
Preţul de desfacere PREŢ
DEPOZITE
Codul depozit CODD
Denumirea depozit DEND
Numărul de salariaţi NRSAL
SALARIAŢI
Marca MARCA
Numele NUME
Funcţia FUNCT
Codul depozitului CODD
Salariul SALA
Codul superiorului CODS
CLIENŢI
Codul clientului CODC
Denumirea clientului DENC
Localitate LOC
Strada STR
Nr. NR
Contul curent CONT
Telefonul TEL
Telefaxul TFAX

Figura 1.1

1.2. Nume şi cuvinte rezervate

Numele unei baze de date, al unei tabele, coloane sau variabile utilizator trebuie să aibă lungimea între 1 şi 30 caractere. Un nume nu poate conţine apostrofuri. Cu atît mai puţin, un nume utilizat într-o comandă nu va fi introdus între apostrofuri. Literele mici şi mari sînt echivalente (nu se face distincţia între literele mici şi mari). Un nume trebuie: să înceapă cu o literă; să conţină numai caracterele A-Z,0-9,$,#,@,-; să difere de un cuvînt rezervat ORACLE; să nu duplice numele unui alt obiect de acelaşi tip.

Cuvintele rezervate nu pot fi utilizate ca nume de tabele, coloane sau orice alte obiecte definite de utilizator. Ele sînt prezentate în *Anexa 1.*

1.3. Operatorii şi funcţiile utilizate în comenzile SQL

Operatorii utilizaţi în comenzile SQL sînt : operatori SQL*PLUS, operatori SQL, operatori aritmetici, operatori logici şi operatori specifici în expresiile de cereri. Ei sînt prezentaţi în *Anexa 2*, în ordinea descrescătoare a priorităţii (întîietăţii), cei de aceeaşi importanţă fiind grupaţi între perechi de linii orizontale. Operatorii sînt evaluaţi de la stînga spre dreapta. Modul lor de utilizare este prezentăt în *paragraful 4.1.b.*

În comenzile SQL*PLUS se utilizează funcţii aritmetice, funcţii caracter, funcţii de grupare, funcţii de conversie, funcţii referitoare la date calendaristice precum şi o serie de pseudo-coloane. Ele sînt prezentate în *Anexa 3* şi exemplificate în *paragraful 4.1.f.*

Scanned with OKEN Scanner

CAPITOLUL 2
CONECTAREA ŞI DECONECTAREA UNEI BAZE DE DATE

Pentru a avea acces la componentele unei baze de date (tabele, vizi-uni, clustere etc.), utilizatorul trebuie mai întîi să se conecteze la ea. La sfîrşitul sau în timpul unei sesiuni SQL există posibilitatea deconectării de la baza de date.

2.1. Deschiderea unei sesiuni de lucru SQL*PLUS

O primă conectare se face atunci cînd se începe lucrul cu SQL*PLUS. Efectul comenzii **SQLPLUS** îl constituie deschiderea unei sesiuni de lucru sub SQL*PLUS.

Comanda SQLPLUS are următoarele forme sintactice:

```
SQLPLUS [nume-utilizator[/parolă]
        [@nume-bază-de-date]
        [@nume-fişier]
        [-SILENT]
SQLPLUS /NOLOG [-SILENT]
SQLPLUS -?
```

unde:

nume-utilizator şi **parolă** sînt numele şi parola unui utilizator cu drept de acces la baza de date.

@nume-bază-de-date este numele unei baze de date cu care se lucrează în reţea (este alt nume decît cel al bazei de date implicite, de pe calculatorul la care se lucrează).

@nume-fişier reprezintă numele unui fişier de comenzi SQL care va fi rulat de SQL*PLUS.

-SILENT are ca efect inhibarea facilităţii de afişare a tuturor informaţiilor şi mesajelor furnizate de SQL*PLUS

/NOLOG lansează în execuţie SQL*PLUS dar nu face conectarea la o bază de date.

Scanned with OKEN Scanner

-? are ca efect afişarea versiunii curente a componentei SQL*PLUS, după care returnează controlul sistemului de operare.

Observaţie: Dacă în linia de comandă se specifică **parola** şi **@nume-bază-de-date** atunci între ele nu trebuie să existe spaţiu. **Parola** sau **@nume-bază-de-date** vor fi separate printr-un spaţiu de **@nume-fişier**.

Exemple:

1) SQLPLUS COMBAZA1/UTIL

SQL*PLUS:Version 2.1 - Production
Connected to: ORACLE V5.1.B - Production

2) SQLPLUS COMBAZA1/UTIL@BIBLIOTEC

SQL*PLUS:Version 2.1 - Production
Connected to: ORACLE V5.1.B - Production

3) SQLPLUS COMBAZA1/UTIL @FISIER.SQL

SQL*PLUS:Version 2.1 - Production
Connected to: ORACLE V5.1.B - Production

4) SQLPLUS COMBAZA1/UTIL -SILENT

5) SQLPLUS -?

SQL*PLUS:Version 2.1 - Production
C:\ORACLE5\BIN>

2.2. Conectarea utilizatorului la o bază de date

Dacă în timpul unei sesiuni de lucru SQL*PLUS se doreşte conectarea la o altă bază de date decît cea deschisă iniţial se poate folosi comanda CONNECT.

Sintaxa acestei comenzi este:

Scanned with OKEN Scanner

```
CONN[ECT] [nume-utilizator[/parolă]]
          [@nume-bază-de-date];
```

unde:

nume-utilizator şi **parolă** sînt numele unui utilizator cu drepturi de acces la baza de date şi parola unui astfel de utilizator; chiar dacă nu se specifică parola (în linia de comandă este opţională), ea este cerută de sistem printr-un mesaj explicit. Parola introdusă la mesajul sistemului va fi invizibilă.

@nume-bază-de-date se specifică în cazul lucrului în reţea, cînd se doreşte conectarea la o altă bază de date decît cea aflată pe calculatorul la care se lucrează.

De remarcat faptul că în timpul unei sesiuni de lucru cu SQL*PLUS se poate realiza conectarea la o bază de date, fără a mai fi necesară închiderea sesiunii.

Furnizarea parametrilor **nume-utilizator** şi **parolă** asigură protecţia bazei de date împotriva accesului neautorizat.

Exemplu:

```
SQL> CONN COMBAZA2/UTIL;
Connected.
```

2.3. *Deconectarea utilizatorului de la baza de date*

Deconectarea utilizatorului de la o bază de date se realizează prin comanda DISCONNECT. Sintaxa acestei comenzi este:

```
DISC[ONNECT];
```

Comanda are ca efect deconectarea de la baza de date curentă, fără a închide sesiunea de lucru SQL*PLUS (pentru aceasta se folosesc comenzile **QUIT, EXIT** sau **^Z**).

Scanned with OKEN Scanner

Exemple:

1) SQL> DISCONNECT;
   Disconnected from ORACLE.
2) SQL> DISC;
   Disconnected from ORACLE.

2.4. Închiderea sesiunii de lucru SQL*PLUS

Închiderea sesiunii de lucru SQL*PLUS şi predarea controlului sistemului de operare al calculatorului gazdă se realizează cu una din comenzile **QUIT, EXIT** sau **^Z**. Sintaxa acestor comenzi este:

```
QUIT;
EXIT;
^Z
```

Exemple:

1) SQL> QUIT;
Disconnected from ORACLE.
C:\ORACLE5\BIN>
2) SQL> EXIT;
Disconnected from ORACLE.
C:\ORACLE5\BIN>
3) SQL> ^Z
Disconnected from ORACLE.
C:\ORACLE5\BIN>

Scanned with OKEN Scanner

CAPITOLUL 3
EDITAREA UNUI PROGRAM SQL

După tastare, o comandă SQL este memorată într-o zonă numită "buffer-ul comenzilor SQL". Această comandă, cunoscută şi sub numele de comandă curentă, rămîne în buffer pînă la tastarea unei alte comenzi SQL.

O facilitate a sistemului de gestiune a bazelor de date ORACLE este aceea că permite modificarea comenzii curente cu ajutorul comenzilor de editare, fără a mai fi astfel nevoie de retastarea întregii comenzii.

3.1. Introducerea liniilor de comandă

Adăugarea unor linii noi după linia curentă (marcată cu un asterisc în partea stîngă) se realizează cu comanda:

```
I[NPUT] [text]
```

Dacă se doreşte introducerea textului după o altă linie decît cea curentă, se poziţionează mai întîi cursorul pe linia respectivă (cu comanda **LIST**) după care se utilizează cuvîntul **INPUT**. Pentru o singură linie, textul trebuie introdus imediat ce a fost lăsat un spaţiu după cuvîntul INPUT. Pentru a introduce mai multe linii se utilizează comanda INPUT fără text, efectul constînd în afişarea numărului liniei curente şi aşteptarea introducerii textului.

Exemplu:

Să se adauge clauza ORDER BY în următoarea comandă SQL:

```
SQL> SELECT MARCA,NUME
  2  FROM SALARIAŢI;
SQL> L2
  2  FROM SALARIAŢI
SQL> INPUT ORDER BY MARCA
```


Scanned with OKEN Scanner

```
SQL> L3
  3  ORDER BY MARCA
```

La relansarea comenzii reţinută în buffer, este vizualizată comanda modificată. Linia modificată este evidenţiată prin marcarea ei cu un asterisc:

```
SQL> RUN
  1  SELECT MARCA,NUME
  2  FROM SALARIATI
  3* ORDER BY MARCA
```

3.2. Adăugarea unui text

Adăugarea unui text la sfîrşitul liniei curente, fără nici un spaţiu între ultimul caracter al liniei şi primul caracter al textului se poate realiza cu ajutorul următoarei comenzi:

```
A[PPEND] text;
```

Pentru ca textul să apară după un spaţiu faţă de ultimul caracter al liniei curente, între cuvîntul APPEND şi text se lasă două spaţii sau se tastează unul singur, iar textul este pus între apostrofuri.

Dacă ultimul caracter din text este caracterul ;, atunci comanda se va termina cu încă un ; (ultima apariţie a acestui caracter fiind interpretată ca delimitator de comandă).

Exemplu:

La linia 1 din programul exemplificat anterior, să se adauge FUNCT şi SALA+VENS:

```
SQL> L1
  1* SELECT MARCA,NUME
```


Scanned with OKEN Scanner

```
SQL> APPEND ,FUNCT,SALA+VENS
  1* SELECT MARCA,FUNCT,SALA+VENS
```

La relansarea în execuţie (cu ajutorul comenzii RUN) este afişat programul modificat:

```
SQL> RUN
  1  SELECT MARCA,NUME,FUNCT,SALA+VENS
  2  FROM SALARIATI
  3* ORDER BY MARCA
```

3.3. Înlocuirea unui text vechi cu unul nou

Dacă se doreşte înlocuirea unui text vechi cu unul nou în linia curentă, se utilizează comanda:

C[HANGE] /text-vechi/text-nou/

Sistemul ignoră tipul caracterelor (litere mari sau mici) din cadrul textului căutat.

Dacă şirul '...' precede o secvenţă de caractere, se vor înlocui caracterele întîlnite pînă la aceasta, inclusiv ea. Dacă acest şir succede secvenţa de caractere vor fi înlocuite toate caracterele care îi urmează, inclusiv ea. Dacă şirul '...' este cuprins între două caractere ale textului, va fi înlocuit textul cuprins între aceste două caractere.

Exemplu:

Presupunînd că linia curentă este: **ORDER BY MARCA**, să se înlocuiască **MARCA** cu **CODD**.

```
SQL> L3
  3* ORDER BY MARCA
SQL> CHANGE/MARCA/CODD
```

Rezultatul va fi afişat:

```
  3* ORDER BY CODD
```


Scanned with OKEN Scanner

Odată cu relansarea în execuţie este afişată şi comanda modificată:

```
SQL> RUN
  1  SELECT MARCA,NUME,FUNCT,SALA+VENS
  2  FROM SALARIATI
  3* ORDER BY CODD
```

3.4. Ştergerea unei linii curente

Ştergerea unei linii curente din program se face cu comanda:

DEL

Exemplu:
Să se şteargă linia a treia din buffer-ul curent.

1) Se poziţionează linia 3 cu ajutorul comenzii **LIST.**

```
SQL> L3
  3* ORDER BY CODD
```

2) Se lansează comanda de ştergere.

```
SQL> DEL
```

3) Se lansează în execuţie programul în noua sa formulare:

```
SQL> RUN
  1  SELECT MARCA,NUME,FUNCT,SALA+VENS
  2* FROM SALARIATI
```

3.5. Listarea liniilor din buffer-ul curent

Pentru listarea liniilor din buffer-ul curent se utilizează comanda:

L[IST][{nr-linie-început|*}|[nr-linie-sfîrşit|*]]

Această comandă are ca efect listarea liniilor din buffer-ul curent cuprinse între cele două numere de linie. Asteriscul indică linia curentă. Buffer-ul curent poate fi schimbat cu ajutorul comenzii **SET BUFFER.**

Scanned with OKEN Scanner

Exemple:
1) Să se listeze conţinutul bufferului SQL.

```
SQL> L

  1  SELECT  MARCA,NUME,FUNCT,SALA+VENS
  2 * FROM SALARIATI
```

2) Să se listeze prima linie din programul SQL.

```
SQL> L1

  1* SELECT MARCA, NUME, FUNCT,SALA+VENS
```

3.6. Salvarea programului din buffer-ul curent

Pentru a salva conţinutul buffer-ului curent într-un fişier recunoscut de sistemul de operare se foloseşte comanda:

SAV[E] nume-fişier
[CRE[ATE] | REP[LACE] | APP[END]];

Dacă tipul fişierului nu este specificat, implicit se consideră SQL.

Prin **CRE[ATE]** se furnizează un mesaj de eroare în cazul în care mai există un fişier cu acelaşi nume (nume-fişier).

REP[LACE] determină înlocuirea vechiului conţinut al fişierului cu unul nou.

Prin **APP[END]** se adaugă la fişierul deja existent conţinutul buffer-ului.

Exemplu:
Să se salveze conţinutul buffer-ului curent în fişierul cu numele **EXEMPLU.SQL**.

```
SQL> L
  1 SELECT MARCA,NUME,FUNCT,SALA+VENS
  2* FROM SALARIATI
SQL> SAVE EXEMPLU.SQL
```


Scanned with OKEN Scanner

3.7. Încărcarea conţinutului unui fişier de tip SQL în buffer-ul curent

Pentru a încărca conţinutul unui fişier de tip SQL în buffer-ul curent se foloseşte comanda:

GET nume-fişier [LIS[T] | NOL[IST]];

Dacă tipul fişierului nu este specificat, el este presupus implicit ca fiind SQL.

Opţiunea **LIS[T]** determină listarea buffer-ului după încărcarea fişierului iar **NOL[IST]** va suprima listarea.

Exemplu:
Să se încarce fişierul EXEMPLU.SQL în buffer-ul curent.

```
SQL> GET EXEMPLU.SQL
  1  SELECT MARCA,NUME,FUNCT,SALA+VENS
  2* FROM SALARIAŢI
```

3.8. Lansarea în execuţie a unui alt editor

Editorul ce urmează a fi lansat este specificat prin variabila utilizator **EDITOR**. Valoarea acestei variabile poate fi schimbată prin comanda **DEFINE**.

În vederea lansării în execuţie a editorului de texte (editor standard) aflat pe calculatorul gazdă şi definit prin comanda **DEFINE** se utilizează comanda:

EDIT [nume-fişier[.extensie]]

unde:

nume-fişier este numele fişierului ce urmează a fi editat. Implicit, tipul fişierului este **SQL**.

În cazul în care nu se specifică nici numele fişierului şi nici extensia sa, editorul va fi apelat pentru programul din buffer-ul curent.

Comanda **EDIT** nu are efect dacă fişierul specificat nu există sau dacă buffer-ul curent este gol.

Scanned with OKEN Scanner

CAPITOLUL 4

SELECTAREA DATELOR DINTR-O BAZĂ DE DATE

Limbajul SQL*PLUS, sub sistemul de gestiune ORACLE, prin intermediul comenzilor sale, oferă facilităţi deosebite privind selectarea datelor din tabelele unei baze de date locale sau aflată într-un nod oarecare dintr-o reţea de calculatoare. Astfel, există posibilitatea selectării unor coloane, linii sau elemente dintr-o tabelă sau, simultan, din mai multe tabele, selectării datelor dintr-un arbore plecînd din orice nod al lui etc. Aceste operaţii pot fi realizate cu efort minim de programare, structura comenzilor fiind apropiată de limbajul natural.

4.1. Comanda SELECT

Pentru a selecta datele din una sau mai multe tabele se utilizează comanda SELECT a cărei sintaxă este:

```
SELECT [ALL | DISTINCT]
 {[nume-tabelă.]* |
      expr[sinonim],expr[sinonim],...}
 FROM nume-tabelă[@legătură][sinonim],
      nume-tabelă[@legătură][sinonim],...
 [WHERE condiţie]
 [CONNECT BY condiţie [START WITH condiţie]]
 [GROUP BY expr,expr,...] [HAVING condiţie]
 [{UNION | INTERSECT | MINUS} SELECT...]
 [ORDER BY {expr | număr-poziţie} [ASC | DESC]
       {expr | număr-poziţie} [ASC | DESC],...
 [FOR UPDATE OF nume-col,
       nume-col,...[NOWAIT]];
```

Clauzele comenzii trebuie utilizate în ordinea specificată în sintaxă, excepţie facînd clauzele CONNECT BY, START WITH, GROUP BY

Scanned with OKEN Scanner

și **HAVING** (care pot fi specificate în orice ordine). Clauzele **ORDER BY** și **FOR UPDATE OF** pot fi schimbate între ele.

Clauza **ALL** determină afișarea tuturor rîndurilor rezultate în urma cererii, spre deosebire de clauza **DISTINCT** care determină eliminarea duplicatelor, afișînd doar rîndurile distincte.

Utilizarea caracterului asterisc (*) are ca efect selectarea tuturor coloanelor din tabela specificată prin clauza **FROM**, în ordinea în care au fost definite la creare. În situația finală, fiecare expresie formulată în comandă devine un nume de coloană. Totodată, orice sinonim, dacă este specificat, este folosit în scopul etichetării expresiei precedente din tabela afișată. Dacă sinonimul conține blancuri sau caractere speciale cum sînt + și -, trebuie incluse între apostrofuri. Pentru a evita ambiguitatea, în cazul în care tabelele conțin coloane cu același nume, este necesară calificarea coloanelor cu numele tabelei (**nume-tabelă**).

Identificarea tabelelor în care trebuie căutate datele corespunzătoare coloanelor specificate se face prin specificarea numelor lor după clauza **FROM**. În locul numelor de tabele se pot folosi sinonimele acestora.

Clauza **WHERE** specifică o condiție care este folosită pentru a selecta rîndurile.

Clauza **CONNECT BY** indică faptul că rîndurile formează o structură arborescentă. Prin această clauză sînt definite relațiile necesare pentru a conecta rîndurile tabelei într-un arbore. Operatorul **PRIOR** folosit înaintea uneia din cele două părți ale condiției, definește nodul părinte iar în cealaltă parte nodul fiu.

Clauza **START WITH**, prin specificarea unei condiții care trebuie satisfăcută, stabilește rîndul folosit ca rădăcină a arborelui. Dacă se omite, comanda **SELECT** va returna o serie de arbori începînd cu fiecare rînd selectat.

Existența clauzei **CONNECT BY** într-o comandă **SELECT** permite utilizarea pseudocoloanei **LEVEL**, care returnează valoarea 1 pentru nodul rădăcină, 2 pentru fiii nodului rădacină, 3 pentru nepoți etc.

Clauzele **GROUP BY** și **HAVING** determină afișarea unor informații sintetice despre grupuri de rînduri care au aceeași valoare în una sau mai multe coloane. Aceste coloane sînt, în general, funcții de grup.

Pentru a combina rezultatele a două comenzi **SELECT** într-un singur rezultat, se folosesc operatorii **UNION**, **INTERSECT** și **MINUS**.

Scanned with OKEN Scanner

UNION returnează rezultatele obţinute de la fiecare cerere în parte.

INTERSECT returnează doar rezultatele comune celor două cereri.

MIŅUS returnează rezultatele obţinute de la prima cerere şi care nu apar în urma celei de a doua selecţii.

Pentru a putea folosi aceste clauze este necesar ca numărul şi tipul coloanelor selectate de fiecare comandă SELECT să fie aceleaşi, lungimile lor putînd fi diferite. Dacă sînt combinate mai mult de două comenzi SELECT, ele vor fi evaluate de la stînga la dreapta.

Pentru a schimba ordinea de evaluare pot fi folosite parantezele. Totodată, cei trei operatori impun utilizarea cuvîntului **DISTINCT** în toate comenzile SELECT.

Clauza **ORDER BY** specifică ordinea în care trebuie returnate rîndurile distincte ale unei tabele.

Clauza **FOR UPDATE OF** determină blocarea rîndurilor selectate ale tabelei astfel încît acestea nu vor mai putea fi actualizate de alţi utilizatori pînă la deblocarea lor cu una din comenzile **COMMIT** sau **ROLL BACK**.

Comanda **SELECT ... FOR UPDATE OF** trebuie urmată de una sau mai multe comenzi **UPDATE ... WHERE**.

Dacă se foloseşte clauza **NOWAIT**, selecţia este considerată terminată chiar dacă rîndurile selectate de **FOR UPDATE** nu pot fi blocate din cauză că alt utilizator lucrează cu ele.

4.2. Utilizarea clauzei FROM

Pentru a selecta una sau mai multe tabele şi pentru a specifica, eventual, identificatorul proprietarului şi legatura cu reţeaua se foloseşte secvenţa:

SELECT ...
FROM [ident-proprietar] tabela [@LINK],...
...;

Exemple:

1) Să se selecteze toate coloanele din tabela SALARIATI.

Scanned with OKEN Scanner

```
SQL> SELECT * FROM SALARIATI;
MARCA NUME       FUNCT  CODD SALA VENS CODS
-----------------------------------------------------------------
1111 AVRAM ION  VINZATO 100000 21200 1000 1000
1222 BARBU DAN  VINZATO 120000 20750 2000 1000
1000 COMAN RADU SEF DEP 130000 35000 2500 1000
3500 DAN ION     VINZATO 160000 24500 3550 2500
2500 VLAD VASILESEF DEP 160000 36500 1500 2500
3700 MANU DAN    VINZAȚO 160000 27500 2500 2500
2650 VLAD ION    VINZATO 120000 25060 3500 1000
7 records selected.
```

2) Să se selecteze coloanele MARCA, NUME, SALA, VENS din tabela SALARIATI.

```
SQL> SELECT MARCA,NUME,SALA,VENS
  2 FROM SALARIATI;
MARCA NUME          SALA      VENS
-----------------------------------------
1111  AVRAM ION      21200     1000
1222  BARBU DAN      20750     2000
1000  COMAN RADU     35000     2500
3500  DAN ION        24500     350
2500  VLAD VASILE    36500     1500
3700  MANU DAN       27500     2500
2650  VLAD ION       25060     3500
7 records selected.
```

3) Să se selecteze coloana NUME din tabela SALARIATI.

```
SQL> SELECT NUME FROM SALARIATI;
NUME
----------
AVRAM ION
BARBU DAN
COMAN RADU
DAN ION
VLAD VASILE
```


Scanned with OKEN Scanner

MANU DAN
VLÁD ION

7 records selected.

4) Să se selecteze coloana FUNCT din tabela SALARIATI în variantele utilizării şi neutilizării clauzei DISTINCT.

```
SQL> SELECT FUNCT FROM SALARIATI;
FUNCT
----------
VINZATO
VINZATO
SEF DEP
VINZATO
SEF DEP
VINZATO
VINZATO

7 records selected.

SQL>SELECT DISTINCT FUNCT FROM SALARIATI;
FUNCT
----------
VINZATO
SEF DEP

2 records selected.
```

4.3. Utilizarea operatorilor în formularea condiţiilor de selecţie din clauza WHERE

Operatorii SQL*PLUS (vezi ***Anexa2***) pot apare în orice parte a unei comenzi şi au întîietate faţă de orice alt tip de operatori. Operatorii aritmetici şi logici sînt utilizaţi pentru formularea condiţiilor de selecţie.

Exemple:

1) Să se selecteze toate înregistrările privind salariaţii al căror salariu este mai mare de 15000 lei:

Scanned with OKEN Scanner

```
SQL> SELECT * FROM SALARIATI
  2  WHERE SALA>15000;
MARCA NUME      FUNCT CODD SALA VENS CODS
---------------------------------------------------------------------
1111 AVRAM ION   VINZATO 100000 21200 1000 1000
1222 BARBU DAN  VINZATO 120000 20750 2000 100
1000 COMAN RADU  SEF DEP 130000 35000 2500 1000
3500 DAN ION     VINZATO 160000 24500  350 2500
2500 VLAD VASILE SEF DEP 160000 36500 1500 2500
3700 MANU DAN    VINZATO 160000 27500 2500 2500
2650 VLAD ION    VINZATO 120000 25060 3500 1000
7 records selected.
```

2) Să se selecteze coloanele MARCA, NUME şi veniturile totale (SALA+VENS) pentru angajaţii care au un salariu mai mare ca 35.000 lei.

```
SQL> SELECT MARCA,NUME,SALA+VENS
   2  FROM SALARIATI
   3  WHERE SALA>35000;
MARCA NUME          SALA+VENS
-------------------------------------------------
2500  VLAD VASILE     38000
1 record selected.
```

3) Să se selecteze coloanele NUME şi FUNCT pentru salariaţii care au funcţia de vînzător.

```
SQL> SELECT NUME,FUNCT FROM SALARIATI
  2  WHERE FUNCT='VINZATO';
NUME          FUNCT
------------------------------
AVRAM ION     VINZATO
BARBU DAN     VINZATO
DAN ION       VINZATO
MANU DAN      VINZATO
VLAD VASILE   VINZATO
5 records selected.
```

4) Să se selecteze coloanele MARCA, NUME, CODD din tabela SALARIATI, dacă venitul suplimentar este mai mare decît salariul.

Scanned with OKEN Scanner

```
SQL> SELECT MARCA,NUME,CODD
  2  FROM SALARIATI
  3  WHERE VENS>SALA;
no records selected.
```

5) Să se selecteaze şi afişeze cîmpurile MARCA,NUME,SALARIU pentru salariaţii ale căror venituri suplimentare sînt mai mari ca 1.500 lei şi lucrează în subordinea superiorului cu codul 1000.

```
SQL> SELECT MARCA,NUME SALA
  2  FROM SALARIATI
  3  WHERE VENS>1500
  4  AND CODS=1000;
MARCA NUME          SALA
-----------------------------------------
1222  BARBU DAN     20750
1000  COMAN RADU    35000
2 records selected.
```

6) Să se afişeze toate coloanele pentru salariaţii cu funcţia vînzător, care au salariul mai mare ca 20.000 lei şi lucrează în subordinea superiorului cu codul 1000.

```
SQL> SELECT * FROM SALARIATI
  2  WHERE FUNCT='VINZATO'
  3  AND SALA>20000
  4  AND CODS=1000;
MARCA NUME      FUNCT CODD SALA VENS CODS
-------------------------------------------------------------------------
1111  AVRAM ION VINZATO 100000 21200 1000 1000
1222  BARBU DAN VINZATO 120000 20750 2000 1000
2 records selected.
```

7) Să se afişeze coloana NUME pentru angajaţi care au salariul mai mic ca 21.000 lei.

```
SQL> SELECT NUME FROM SALARIATI
  2  WHERE SALA<30000;
NUME
----------
AVRAM ION
```


Scanned with OKEN Scanner

BARBU DAN
DAN ION
MANU DAN
VLAD ION
5 records selected.

8) Să se selecteze înregistrările pentru care funcţia este 'SEF DEP' sau salariul este mai mare decît 35.000 lei.

```
SQL> SELECT MARCA,CODS FROM SALARIATI
  2  WHERE FUNCT='SEF DEP' OR SALA>35000;
MARCA           CODS
--------------- ---------------
1000            1000
2500            2500
2550            2500
3 records selected.
```

9) Să se selecteze datele despre salariaţii care au funcţia de vînzător şi nu lucrează în subordinea superiorului cu codul 1000.

```
SQL> SELECT * FROM SALARIATI
  3    WHERE FUNCT='VINZATO'
  2    AND CODS!=1000;
MARCA NUME    FUNCT   CODD  SALA VENS CODS
-------------------------------------------------------------------------
3500 DAN ION    VINZATO 160000 24500  750 2500
3700 MANU DAN VINZATO 160000 27500 2500 2500
2 records selected.
```

10) Să se selecteze toţi salariaţii care lucrează în subordinea superiorului cu codul 1000 precum şi cei care au salariul mai mic de 30000 lei sau funcţia de vînzător.

```
SQL> SELECT *
  2  FROM SALARIATI
  2  WHERE FUNCT='VINZATO' OR SALA<30000
  3  AND CODS=1000;
MARCA NUME     FUNCT   CODD SALA VENS CODS
-------------------------------------------------------------------------
1111 AVRAM ION VINZATO 100000 21200 1000 1000
1222 BARBU DAN VINZATO 120000 20750 2000 1000
```


Scanned with OKEN Scanner

```
3500 DAN ION     VINZATO 160000 24500  350 2500
3700 MANU DAN  VINZATO 160000 27500 2500 2500
2650 VLAD ION   VINZATO 120000 25060 3500 1000
5 records selected.
```

11) Să se selecteze toate înregistrările care conţin date despre angajaţii a căror funcţie este cea de şef de depozit sau care au un salariu de 35000 de lei şi lucrează în subordinea superiorului cu codul 1000.

```
SQL> SELECT MARCA,NUME
  2  FROM SALARIATI
  3  WHERE FUNCT='SEF DEP' OR
  4        (SALA=35000 AND
  5     CODS=1000);
MARCA NUME
----- ---------------
1000 COMAN RADU
2500 VLAD VASILE
2550 FRINCU ION
3 records selected.
```

12) Să se selecteże marca şi numele salariaţilor care lucrează în subordinea superiorului cu marca 1000 şi au funcţia şef de depozit sau salariul în valoare de 35.000 de lei

```
SQL> SELECT MARCA,NUME FROM SALARIATI
  2  WHERE (FUNCT='SEF DEP' OR SALA=35000)
  3  AND CODS=1000;
MARCA  NUME
------ ---------------
1000 COMAN RADU
```

13) Să se afişeze valorile coloanelor MARCA, NUME, FUNCT privind angajaţii care lucrează în subordinea superiorului cu marca 1000 şi au funcţia de şef de depozit sau de vînzător.

```
SQL> SELECT MARCA,NUME,FUNCT
  2  FROM SALARIATI
  3  WHERE (FUNCT='SEF DEP' OR
  4        FUNCT='VINZATO')
  5  AND CODS=1000;
```


Scanned with OKEN Scanner

```
MARCA NUME                FUNCT
-----------------------------------------
1111  AVRAM ION           VINZATO
1222  BARBU DAN           VINZATO
1000  COMAN RADU          SEF DEP
```

14) Să se selecteze coloanele MARCA, NUME, FUNCT pentru salariaţii care au funcţia de şef depozit sau pentru cei care lucrează în subordinea superiorului cu marca 1000 şi au funcţia de vînzător.

```
SQL> SELECT MARCA,NUME,FUNCT
  2  FROM SALARIATI
  3  WHERE FUNCT='SEF DEP'
  4  OR (FUNCT='VINZATO'
  5  AND CODS=1000);
MARCA NUME                FUNCT
-----------------------------------------
1111 AVRAM ION            VINZATO
1222 BARBU DAN            VINZATO
1000 COMAN RADU           SEF DEP
2650 VLAD ION             VINZATO
2500 VLAD VASILE          SEF DEP
2550 FRINCU ION           SEF DEP
6 records selected.
```

15) Să se selecteze toate datele privind angajaţii ce nu au funcţia de vînzător.

```
SQL> SELECT * FROM SALARIATI
  2  WHERE NOT(FUNCT='VINZATO');
MARCA NUME       FUNCT CODD SALA VENS CODS
-------------------------------------------------------------------------
1000 COMAN RADU  SEF DEP 13000 35000  2500 1000
2500 VLAD VASILE SEF DEP 16000 36500  1500 2500
2550 FRINCU ION  SEF DEP 16000 36000 37000 2500
3 records selected.
```

16) Să se selecteze valorile coloanelor NUME, MARCA, FUNCT, SALA+VENS pentru angajaţii care au salariul cuprins între 24500 şi 36000 lei.

Scanned with OKEN Scanner

```
SQL> SELECT NUME,MARCA,FUNCT,SALA+VENS
 .2  FROM SALARIATI
  3  WHERE SALA BETWEEN 24500 AND 36000;
NUME       MARCA FUNCT  SALA+VENS
-------------------------------------------------------
COMAN RADU  1000 SEF DEP 37500
DAN ION       3500 VINZATO 24850
MANU DAN     3700 VINZATO 30000
FRINCU ION  2550 SEF DEP  73000
VLAD ION      2650 VINZATO 25060
5 records selected.
```

17) Să se selecteze cîmpurile NUME, MARCA, FUNCT, SALA+VENS pentru salariaţii care au salariul mai mic decît 24500 şi mai mare decît 36000.

```
SQL> SELECT NUME,MARCA,FUNCT,SALA+VENS
  2  FROM SALARIATI
  3  WHERE SALA NOT BETWEEN 24500 AND 36000;
NUME       MARCA FUNCT   SALA+VENS
-------------------------------------------------------
AVRAM ION    1111  VINZATO 22200
BARBU DAN    1222  VINZATO 22750
VLAD ION      2650  VINZATO 25060
VLAD VASILE 2500  SEF DEP  38000
4 records selected.
```

18) Să se selecteze cîmpurile MARCA,NUME, FUNCT pentru salariaţii care lucrează în depozitul cu codurile 100000 sau 160000.

```
SQL> SELECT MARCA,NUME,FUNCT
  2  FROM SALARIATI
  3  WHERE CODD IN(100000,160000);
MARCA NUME            FUNCT
-----------------------------------------
1111 AVRAM ION        VINZATO
2650 VLAD ION         VINZATO
3500 DAN ION          VINZATO
2500 VLAD VASILE      SEF DEP
```


Scanned with OKEN Scanner

```
3700 MANU DAN          VINZATO
2550 FRINCU ION        SEF DEP
6 records selected.
```

19) Să se selecteze cîmpurile MARCA, NUME, SALA, VENS pentru salariaţii care au altă funcţie decit cea de vînzător.

```
SQL> SELECT MARCA,NUME,SALA,VENS
  2 FROM SALARIATI
  3 WHERE FUNCT NOT IN ('VINZATO');
MARCA NUME          SALA          VENS
-----------------------------------------------
1000 COMAN RADU     35000         2500
2500 VLAD VASILE    36500         1500
2550 FRINCU ION     36000         37000
```

Operatorul LIKE face posibilă realizarea unor cereri de regăsire a înregistrărilor ce conţin un cîmp a cărui valoare este comparată cu un şir de caractere dat.

Pentru a evalua expresiile logice în care apar şiruri de caractere se poate folosi clauza **LIKE** în următoarea secvenţă:

```
SELECT ...
    WHERE coloană LIKE şir
  ...;
```

Avantajul unei viteze mari de regăsire ca urmare a indexării este pierdut în momentul în care se caută într-o coloană indexată un şir care începe cu - sau %. Aceste caractere speciale suplinesc unul, respectiv mai multe caractere.

Exemple:

1) Să se selecteze coloanele NUME, MARCA, FUNCT şi veniturile totale pentru salariaţii al căror nume începe cu litera C.

```
SQL> SELECT NUME,MARCA,FUNCT,SALA+VENS
  2 FROM SALARIATI
  3 WHERE NUME LIKE 'C%';
NUME          MARCA FUNCT     SALA+VENS
-----------------------------------------------
COMAN RADU  1000  SEF DEP     37500
```


Scanned with OKEN Scanner

2) Să se selecteze coloanele NUME,MARCA,FUNCT,SALA+VENS pentru salariaţii al căror nume se termină cu litera N.

```
SQL> SELECT NUME,MARCA,FUNCT,SALA+VENS
  2  FROM SALARIATI
  3  WHERE NUME LIKE '%N';

NUME         MARCA FUNCT  SALA+VENS
-------------------------------------------------------------
AVRAM ION    1111  VINZATO 22200
BARBU DAN    1222  VINZATO 22750
DAN ION      3500  VINZATO  24850
MANU DAN     3700  VINZATO 30000
FRINCU ION   2550  SEF DEP  73000
VLAD ION     2650  VINZATO 28560
```

3) Să se selecteze coloanele NUME, MARCA, FUNCT, SALA+VENS pentru salariaţii al căror nume este format din nouă caractere, pe ultima poziţie fiind N.

```
SQL> SELECT NUME,MARCA,FUNCT,SALA+VENS
  2  FROM SALARIATI
  3  WHERE NUME LIKE '_ _ _ _ _ _ _ _N'

NUME          MARCA FUNCT  SALA+VENS
-------------------------------------------------------------
AVRAM ION     1111  VINZATO 22200
BARBU DAN     1222  VINZATO 22750
```

4) Să se selecteze salariaţii al căror nume are pe poziţia a treia litera M.

```
SQL> SELECT NUME,MARCA,FUNCT,SALA+VENS
  2  FROM SALARIATI
  3  WHERE NUME LIKE '_ _M%'

NUME         MARCA FUNCT   SALA+VENS
-------------------------------------------------------------
COMAN RADU   1000  SEF DEP 37500
```

5) Să se selecteze salariaţii al căror nume are o lungime de nouă caractere.

```
SQL> SELECT NUME,MARCA,FUNCT,SALA+VENS
  2  FROM SALARIATI
```


Scanned with OKEN Scanner

```
  3  WHERE NUME LIKE '_ _ _ _ _ _ _ _ _'
NUME          MARCA FUNCT   SALA+VENS
-------------------------------------------------------
AVRAM ION     1111  VINZATO 22200
BARBU DAN     1222  VINZATO 22750
```

Regăsirea unor înregistrări ce conţin cîmpuri cu valori nule se face cu ajutorul **operatorului NULL.**

Exemple:

1) Să se selecteze cîmpurile MARCA,NUME,FUNCT pentru salariaţii ce nu au venit suplimentar.

```
SQL> SELECT MARCA,NUME,FUNCT
  2  FROM SALARIATI
  3  WHERE VENS IS NULL;
MARCA NUME       FUNCT  CODD SALA VENS CODS
-------------------------------------------------------------------
3770 CARMEN ANA VINZATO 130000 26500      4000
```

Observaţie: O înregistrare cu un cîmp ce conţine o valoare nulă este introdusă astfel:

```
SQL> INSERT INTO SALARIATI VALUES
  2 (3770,'CARMEN ANA','VINZATO',130000,
  3  26500,NULL,4000)
1 record created.
```

2) Să se selecteze toate înregistrările al căror cîmp VENS (venituri suplimentare) nu conţine valoarea NULL.

```
SQL> SELECT * FROM SALARIATI WHERE
  2 VENS IS NOT NULL;
MARCA NUME  FUNCT   CODD   SALA VENS  CODS
-------------------------------------------------------------------
1111 AVRAM ION  VINZATO 100000 21200 1000 1000
1222 BARBU DAN VINZATO 120000 20750 2000 1000
1000 COMAN RADU SEF DEP 130000 35000 2500 1000
                    .....
10 records selected.
```


Scanned with OKEN Scanner

4.4. Ordonarea liniilor rezultate în urma unei cereri

Limbajul SQL*PLUS are posibilitatea ordonării crescătoare sau des-crescătoare a liniilor rezultate în urma unei cereri. Această operaţie se realizează prin intermediul secvenţei:

```
SELECT...
   FROM ...
   WHERE ...
   ORDER BY {expr | număr-poziţie}[ASC | DESC],...
  ...;
```

unde:

expr reprezintă o expresie care face referire la una sau mai multe coloane;

număr-poziţie este un număr care identifică poziţia coloanei din comanda SELECT, după care se doreşte sortarea. Utilizarea operatorului de tipul **UNION, INTERSECT** sau **MINUS** impune prezenţa argumentului **număr-poziţie.**

ASC sau **DESC** precizează modul de ordonare ascendent, respectiv descendent.

În cazul în care clauzele **ORDER BY** şi **DISTINCT** sînt utilizate împreună, clauza **ORDER BY** trebuie să se refere la coloane care n-au fost menţionate în comanda SELECT.

Exemple:

1) Să se selecteze toate coloanele tabelei SALARIATI privind salariaţii care sînt vînzători şi pentru care marca superiorului este 1000. Afişarea să se facă după valorile coloanei MARCA, des-crescător.

```
SQL> SELECT * FROM SALARIATI
  2  WHERE CODS=1000
  3  AND FUNCT='VINZATO'
  4  ORDER BY MARCA DESC;
MARCA NUME     FUNCT   CODD SALA VENS CODS
-------------------------------------------------------------
2650 VLADA ION VINZATO 120000 25060 3500 1000
```


Scanned with OKEN Scanner

1111 AVRAM ION VINZATO 100000 21200 1000 1000
1222 BARBU DAN VINZATO 120000 20750 2000 1000
3 records selected.

2) Să se selecteze crescător după salariu, angajaţii cu funcţia vînzător, şi pentru care marca superiorului este 1000.

```
SQL> SELECT * FROM SALARIATI
  2  WHERE CODS=1000
  3  AND FUNCT='VINZATO'
  4  ORDER BY SALA;
```

```
MARCA NUME     FUNCT    CODD SALA VENS CODS
-------------------------------------------------------------------
1222 BARBU DAN VINZATO 120000 20750 2000 1000
1111 AVRAM ION VINZATO 100000 21200 1000 1000
2650 VLAD ION   VINZATO 120000 25060 3500 1000
3 records selected.
```

3) Să se selecteze crescător, după salariu, coloanele MARCA, NUME, SALA, SALA+VENS pentru acei salariaţi cu funcţia vînzător şi pentru care marca superiorului este 1000,

```
SQL> SELECT MARCA,NUME,SALA,VENS
 2  FROM SALARIATI
 3  WHERE CODS=1000 AND FUNCT='VINZATO'
 4  ORDER BY SALA;
```

```
MARCA NUME     SALA  VENS
-----------------------------------------
1222 BARBU DAN 20750 2000
1111 AVRAM ION  21200 1000
2650 VLAD ION    25060 3500
3 records selected.
```

4) Să se selecteze coloanele MARCA, NUME, CODD, VENS ordonate crescător după codul depozitului şi veniturile suplimentare.

```
SQL> SELECT MARCA,NUME,CODD,VENS
  2  FROM SALARIATI
  3  WHERE FUNCT='VINZATO'
  4  ORDER BY CODD,VENS;
```


Scanned with OKEN Scanner

```
SQL> SELECT CONTRACT.CODP,
  2    CONTRACT.CANT,CONTRACT.PRET,DENP,
  3    COS.STOC
  4  FROM COMENZI CONTRACT,PRODUSE COS
  5  WHERE CONTRACT.CODP=COS.CODP
  6    AND UM='BUC'
  7  ORDER BY CONTRACT.CANT*COS.PRET
CODP   CANT          PRET DENP       STOC
-------------------------------------------------------------
14444  6             4500 SCAUN D4    .   36
13333  4            80000 CANAPEA A7      6
2 records selected.
```

3) Să se selecteze, în condiţiile redenumirii tabelelor COMENZI cu *CONTRACT* şi PRODUSE cu *COS*, a coloanelor CODP, CANT, PRET din CONTRACT şi coloanelor DENP, STOC din COS. Să fie selectate doar cîmpurile care au unitatea de măsură **buc**. Liniile să fie ordonate crescător după valorile comenzilor. La afişare să se extragă şi o coloană numită 'DIFERENTA' care să reflecte stocul rămas în urma onorării comenzii.

```
UFI> SELECT CONTRACT.CODP,
  2    CONTRACT.CANT,CONTRACT.PRET,
  3    DENP,COS.STOC,
  4    COS.STOC-CONTRACT.CANT DIFERENTA
  5  FROM COMENZI CONTRACT,PRODUSE COS
  6    WHERE CONTRACT.CODP=COS.CODP
  7    AND UM='BUC'
  8  ORDER BY
  9  CONTRACT.CANT*CONTRACT.PRET
CODP CANT PRET  DENP       STOC DIFERENTA
-------------------------------------------------------------
14444 6     4500 SCAUN D4     36   30
13333 4    80000 CANAPEA A7   6    2
2 records selected.
```

4) Din tabela COMENZI (definită prin etichetele T1 şi T2) să se selecteze CODP, CODC, în condiţiile în care valoarea comenzii este mai mare ca 50.000 lei. (JOIN-ul unei tabele pe ea însăşi).

Scanned with OKEN Scanner

```
SQL> SELECT DISTINCT T1.CODP,T2.CODC
 2  FROM COMENZI T1,COMENZI T2
 3  WHERE T1.CANT*T2.PRET>50000
CODP          CODC
---------     ---------
13333         121111
14444         121111
16666         121111
22222         121111
4 records selected.
```

5) Să se selecteze cîmpurile CODP, DENP, PRET, PRETMAX, PRETMIN pentru produsele al căror preţ negociat este cuprins între preţul maxim şi preţul minim. Ordonarea să se facă crescător după cîmpul CODP.

```
SQL> SELECT PRODUSE.CODP,DENP,
 2        COMENZI.PRET,
 3        PRETMIN,PRETMAX
 4  FROM PRODUSE,PRETURI,COMENZI
 5  WHERE COMENZI.PRET BETWEEN PRETMIN
 6    AND PRETMAX
 7  ORDER BY PRODUSE.CODP
CODP  DENP        PRET     PRETMIN  PRETMAX
-----------------------------------------------------------
13333 CANAPEA A7 80000     79000        82000
14444 SCAUN D4     4500     4500         5000
16666 PLACAJ 2/2  25000    24000        30000
3 records selected.
```

4.6. Realizarea cererilor incluse

Subcererile reprezintă cereri incluse în clauzele unor comenzi SQL.

Rîndurile selectate de o subcerere nu sînt afişate, ele fiind utilizate în continuare de o comandă SQL.

Dacă subcererea este folosită în partea dreaptă a unei expresii logice sau a unei expresii de atribuire, ea va returna o singură valoare sau o coloană de valori.

Scanned with OKEN Scanner

Compunerea valorii rezultate cu cea din stînga expresiei se face în conformitate cu operatorul care face legătura între cele două părţi.

În cazul în care subcererea este folosită pentru a specifica valori în comenzi ca **INSERT, CREATE TABLE, UPDATE**, ea va returna cîte o valoare pentru fiecare coloană specificată în comandă.

Clauze ca **ORDER BY, FOR** nu pot fi folosite în subcereri.

Subcererile apar, în general, în următoarele comenzi:

```
1) COPY [FROM nume-utiliz/parolă@bază-de-date]
        [TO nume-utiliz/parolă@bază-de-date]
        {APPEND | CREATE | INSERT | REPLACE}
        tabelă | (col,col...) | USING cerere;
2) CREATE TABLE tabelă ...
        AS cerere;
3) INSERT INTO tabelă | [(col,col,...)]
        [VALUES (val,val...) | cerere];
4) UPDATE tabelă [sinonim]
        SET (col,col,...)=(cerere)
        [WHERE condiţie];
```

Există si subcereri corelate cu cererile din comanda principală, care apar doar în clauza **WHERE** a comenzii **SELECT**.

Ele pot utiliza sinonime pentru tabela precizată în comanda **SELECT** şi sînt evaluate cîte o dată pentru fiecare rînd selectat în comanda principală. Subcererile corelate pot apare în formule ca:

```
1) SELECT
      coloana1,coloana2,...
      FROM tabela1,tabela2 tab2,...
      WHERE coloana1 IN
        (SELECT coloana1
         FROM tabela1
         WHERE condiţie)
   ...;
2) SELECT
      coloana1,coloana2,...
      FROM tabela1,tabela2 tab2,...
      WHERE coloana2 IN
       (SELECT funcţie(coloana)
          FROM tabela2
          WHERE tab2.coloana=coloana)
   ...;
```


Scanned with OKEN Scanner

Exemple:

1) Să se selecteze cîmpurile NUME şi FUNCT ale salariaţilor cu funcţia identică cu a lui RADU IOANA.

```
SQL> SELECT NUME,FUNCT FROM SALARIATI
  2  WHERE FUNCT=
  3       (SELECT FUNCT FROM SALARIATI
  4        WHERE NUME='RADU IOANA');

NUME                 FUNCT
-------------------- ---------------
AVRAM ION            VINZATO
BARBU DAN            VINZATO
DAN ION              VINZATO
MANU DAN             VINZATO
VLAD ION             VINZATO
SANDU ION            VINZATO
CARMEN ANA           VINZATO
RADU IOANA           VINZATO
ALEXE IOAN           VINZATO
9 records selected.
```

2) Să se selecteze, în modul DISTINCT, valorile cîmpurilor MARCA, SALA, NUME, CODS ale angajaţilor care au salariul mai mare decît unul dintre subordonaţii superiorului cu codul 1000. Rezultatele sînt cerute ordonate descrescător, după valorile cîmpului SALA.

```
SQL> SELECT DISTINCT MARCA,SALA,
  2        NUME,CODS
  3  FROM SALARIATI
  4  WHERE SALA> ANY
  5        (SELECT SALA FROM SALARIATI
  6        WHERE CODS=1000)
  7  ORDER BY SALA DESC;

MARCA  SALA  NUME        CODS
-----------------------------------------
2500   36500 VLAD VASILE  2500
3755   36500 DORU DAN     4000
2550   36000 FRINCU ION   2500
1000   35000 COMAN RADU   1000
```


Scanned with OKEN Scanner

MARCA	NUME	CODD	VENS
1111	AVRAM ION	100000	1000
1222	BARBU DAN	120000	2000
2650	VLAD ION	120000	3500
3500	DAN ION	160000	350
3700	MANU DAN	160000	2500

4.5. Selecţii din mai multe tabele

Operaţia prin care se selectează şi se grupează coloanele din tabele diferite, în scopul obţinerii unor informaţii coerente, poartă numele de joncţiune (**join**).

Pentru a realiza o joncţiune şi pentru a preciza corespondenţa între rîndurile tabelelor, se utilizează urmatoarea secvenţă:

```
SELECT ...
        FROM nume-tab1,nume-tab2,...
  WHERE condiţie
  ...;
```

Unde:

condiţie reprezintă orice expresie care compară cîmpurile diferitelor tabele. Vor fi selectate rîndurile pentru care condiţia este îndeplinită. În **condiţie** poate fi folosit operatorul (+) care simulează existenţa unor rînduri vide în tabelele pentru care nu se găsesc corespondenţe.

Exemple:

1) Să se selecteze codul produsului, cantitatea şi diferenţa dintre valoarea mărfurilor comandate la preţul maxim şi cel efectiv negociat. Ordonarea să se facă ascendent după CODP.

```
SQL> SELECT COMENZI.CODP,CANT,
  2       PRETMAX*CANT-PRET*CANT
  3 FROM COMENZI,PRETURI
  4 WHERE PRETURI.CODP=COMENZI.CODP
  5 ORDER BY COMENZI.CODP;
```


Scanned with OKEN Scanner

```
CODP  CANT   PRETMAX*CANT-PRET*CANT
-----------------------------------------------------
13333  4      8000
14444  6      3000
16666 15     75000
3 records selected.
```

2) Să se afişeze din tabela COMENZI codul produsului, diferenţa dintre valoarea mărfurilor comandate la preţul maxim şi cel efectiv negociat, diferenţa dintre valoarea mărfurilor comandate la preţul minim şi cel efectiv negociat selectate după criteriul egalităţii EQUI-JOIN (JOIN pe condiţie de egalitate). Liniile să fie ordonate crescător după valoarea totală a comenzii.

```
SQL> SELECT COMENZI.CODP,
  2  PRETMAX*CANT-PRET*CANT,
  3  PRETMIN*CANT-PRET*CANT
  4  FROM COMENZI,PRETURI
  5  WHERE PRETURI.CODP=COMENZI.CODP
  6  ORDER BY PRET*CANT;
```

```
CODP PRETMAX*CANT-PRET*CANT                    PRETMIN*
                                          CANT-PRET*CANT
------- ------------------------------------ ------------------------
14444  3000                                             0
13333  8000                                         -4000
16666 75000                                        -15000
3 records selected.
```

3) Să se afişeze cîmpurile CODP, DENP, STOC şi CANT, utilizînd criteriul egalităţii OUTER-JOIN, pe cîmpul comun CODP (se afişează datele despre acele produse pentru care există comenzi dar nu sînt în nomenclatorul de produse). Liniile să fie ordonate crescător după cîmpul CODP.

```
SQL> SELECT COMENZI.CODP,DENP,
  2    STOC,CANT
  3  FROM PRODUSE,COMENZI
  4  WHERE COMENZI.CODP=PRODUSE.CODP(+)
  5  ORDER BY COMENZI.CODP;
```


Scanned with OKEN Scanner

```
CODP   DENP          STOC   CANT
-------------------------------------------
13333 CANAPEA A7       6     4
14444 SCAUN D4        36     5
16666 PLACAJ 2/2     100    15
22222                        12
4 records selected.
```

4) Să se afişeaze în modul DISTINCT (fără a se repeta aceleaşi linii), codul depozitului şi funcţiile care conţin valori nule în cîmpul veniturilor suplimentare. Selectarea să se facă pe criteriul egalităţii pe cîmpul comun CODD şi în condiţiile în care acel depozit există.

```
SQL> SELECT DISTINCT DEPOZITE.CODD,FUNCT
  2  FROM SALARIATI,DEPOZITE
  3  WHERE SALARIATI.CODD=DEPOZITE.CODD(+)
  4  AND VENS IS NULL;
CODD   FUNCT
------------------------
130000 VINZATO
160000 VINZATO
2 records selected.
```

5) Să se afişeze în modul DISTINCT codul depozitului şi fun.[illegible]le care conţin valori nule în cîmpul veniturilor suplimentare. Selectarea se face pe criteriul egalităţii pe cîmpul comun CODD şi în condiţiile în care acel depozit există. Liniile să fie ordonate descrescător după codul depozitului (din tabela SALARIATI).

```
SQL> SELECT DISTINCT DEPOZITE.CODD,FUNCT
  2  FROM SALARIATI,DEPOZITE
  3  WHERE SALARIATI.CODD=DEPOZITE.CODD(+)
  4  AND VENS IS NULL
  5  ORDER BY SALARIATI.CODD DESC
CODD   FUNCT
------------------------
160000 VINZATO
130000 VINZATO
2 records selected.
```


Scanned with OKEN Scanner

Într-o cerere se pot înlocui numele de tabele sau coloane prin etichete.

Exemple:

1) Să se selecteze NUME,FUNCT,DEND din tabelele DEPOZITE (pentru care este utilizat numele D) și SALARIATI (cu numele S). Liniile să fie ordonate crescător după cîmpul NUME.

```
SQL> SELECT NUME,FUNCT,D.CODD,DEND
  2  FROM DEPOZITE D, SALARIATI S
  3  WHERE D.CODD=S.CODD
  4  ORDER BY NUME
NUME          FUNCT      CODD  DEND
-------------------------------------------------
ALEXE IOAN    VINZATO  160000 SPORT
AVRAM ION     VINZATO  100000 MOBILA
BARBU DAN     VINZATC  120000 ALIMENTAR
CARMEN ANA  VINZATO  130000 AUTO
COMAN RADU  SEF DEP   130000 AUTO
DAN ION       VINZATO    160000 SPORT
DORU DAN      SEF DEP    130000 AUTO
FRINCU ION    SEF DEP    160000 SPORT
MANU DAN      VINZATO  160000 SPORT
RADU IOANA  VINZATO  130000 AUTO
SANDU ION     VINZATO  130000 AUTO
VLAD ION      VINZATO  120000 ALIMENTAR
VLAD VASILE SEF DEP   160000 SPORT
13 records selected.
```

Observație: Pentru a pune în evidență posibilitatea schimbării denumirilor de tabele, în exemplele care urmează se va folosi, cu preponderență, adresarea prin calificare.

2) Să se selecteze, în condițiile redenumirii tabelelor COMENZI cu *CONTRACT* și PRODUSE cu *COS*, a coloanelor CODP, CANT, PRET din CONTRACT și coloanelor DENP, STOC din COS. Să fie selectate doar cîmpurile care au unitatea de măsură **'buc'**. Liniile să fie ordonate crescător după valorile comenzilor.

Scanned with OKEN Scanner

```
3700   27500 MANU DAN       2500
3770   26500 CARMEN ANA     4000
3755   25700 ALEXE IOAN     2500
3760   25600 SANDU ION      4000
2650   25060 VLAD ION       1000
3500   24500 DAN ION        2500
1111   21200 AVRAM ION      1000
11 records selected.
```

3) Să se selecteze valorile cîmpurilor MARCA, NUME, SALA, CODS ale angajaţilor care au salariul mai mare decît al oricărui salariat din subordinea angajatului cu codul 1000. Datele să fie ordonate descrescător după valorile cîmpului MARCA.

```
SQL> SELECT MARCA,NUME,SALA,CODS
  2  FROM SALARIATI
  3  WHERE SALA>ALL
  4        (SELECT SALA FROM SALARIATI
  5        WHERE CODS=1000)
  6  ORDER BY MARCA DESC;

MARCA NUME                 SALA       CODS
-------------------------------------------
3755 DORU DAN              36500      4000
2550 FRINCU ION            36000      2500
2500 VLAD VASILE           36500      2500
3 records selected.
```

4) Să se selecteze cîmpurile CODD, DEND, NRSAL pentru acele depozite care au numărul de salariaţi mai mare ca cinci şi codul cuprins în intervalul 100000 ÷ 130000.

```
SQL> SELECT CODD,DEND,NRSAL
  2  FROM DEPOZITE
  3  WHERE NRSAL>5 AND CODD IN
  4  (SELECT CODD FROM DEPOZITE
  5  WHERE CODD BETWEEN 100000 AND 130000);

CODD DEND                  NRSAL
-------------------------------------------
120000 ALIMENTAR           11
1 record selected.
```


Scanned with OKEN Scanner

5) Să se selecteze CODD, DEND, NRSAL pentru acele depozite care au numărul de salariaţi mai mare ca 5 şi codul în afara intervalului 100000 ÷ 130000.

```
SQL> SELECT CODD,DEND,NRSAL
 2  FROM DEPOZITE
 3  WHERE NRSAL>5 AND CODD NOT IN
 4        (SELECT CODD FROM DEPOZITE
 5        WHERE CODD
 6        BETWEEN 100000 AND 130000);
CODD DEND                    NRSAL
-----------------------------------------------
140000 TEXTILE                  7
1 record selected.
```

6) Să se selecteze următoarele informaţii: marca, funcţia şi veniturile totale ale salariaţilor care au funcţia şi salariul lui VLAD VASILE.

```
SQL> SELECT MARCA,NUME,SALA+VENS
 2  FROM SALARIATI
 3  WHERE (FUNCT,SALA)=
 4        (SELECT FUNCT,SALA FROM SALARIATI
 5        WHERE NUME='VLAD VASILE');
MARCA NUME        SALA+VENS
-----------------------------------------
2500  VLAD VASILE 38000
3755  DORU DAN    42000
2 records selected.
```

7) Să se selecteze MARCA, NUME, SALA+VENS pentru acei angajaţi care au funcţia lui VLAD VASILE sau salariul mai mare sau egal cu cel pe care îl are RADU IOANA. Ordonarea este crescătoare după valorile cîmpului NUME.

```
SQL> SELECT MARCA,NUME,SALA+VENS
 2  FROM SALARIATI
 3  WHERE FUNCT IN
 4        (SELECT FUNCT FROM SALARIATI
 5         WHERE NUME='VLAD VASILE')
 6  OR SALA>=
```


Scanned with OKEN Scanner

```
 7        (SELECT SALA FROM SALARIATI
 8         WHERE NUME='RADU IOANA')
 9  ORDER BY NUME;
MARCA NUME                                          SALA+VENS
-----------------------------------------------------------------------
3755 ALEXE IOAN
1111 AVRAM ION                                        22200
1222 BARBU DAN                                        22750
3770 CARMEN ANA
1000 COMAN RADU                                       37500
3500 DAN ION                                          24850
3755 DORU DAN                                         42000
2550 FRINCU ION                                       73000
3700 MANU DAN                                         30000
1680 RADU IOANA                                       23750
3760 SANDU ION                                        25600
2650 VLAD ION                                         28560
2500 VLAD VASILE                                      38000
13 records selected.
```

Observaţie: Dacă se doreşte evitarea afişării valorilor nule (vezi paragraful 4.7.b) ale cîmpului SALA+VENS, se poate proceda astfel:

```
SQL> SELECT MARCA,NUME,SALA+VENS
 2  FROM SALARIATI
 3  WHERE FUNCT IN
 4     (SELECT FUNCT FROM SALARIATI
 5      WHERE NUME='VLAD VASILE')
 6  OR SALA>=
 7        (SELECT SALA FROM SALARIATI
 8        WHERE NUME='RADU IOANA')
 9    AND VENS IS NOT NULL
10  ORDER BY NUME;
MARCA NUME                          SALA+VENS
-----------------------------------------------------
1111 AVRAM ION                        22200
1222 BARBU DAN                        22750
1000 COMAN RADU                       37500
```

```
3500 DAN ION                    24850
3755 DORU DAN                   42000
2550 FRINCU ION                 73000
3700 MANU DAN                   30000
1680 RADU IOANA                 23750
3760 SANDU ION                  25600
2650 VLAD ION                   28560
2500 VLAD VASILE                38000
11 records selected.
```

8) Să se selecteze cîmpurile NUME şi CODD ale angajaţilor cu codul superiorului 1000 şi care au aceeaşi funcţie cu DORU DAN. Să se afişeze numai salariaţii pentru care există corespondenţă de CODD în tabelele DEPOZITE şi SALARIATI. Datele să fie ordonate crescător după valorile cîmpurilor CODD şi NUME.

```
SQL> SELECT NUME,DEPOZITE.CODD
  2  FROM SALARIATI,DEPOZITE
  3  WHERE CODS=1000
  4  AND DEPOZITE.CODD=SALARIATI.CODD
  5  AND FUNCT IN
  6     (SELECT FUNCT FROM SALARIATI
  7      WHERE NUME='DORU DAN')
  8  ORDER BY DEPOZITE.CODD,NUME;

NUME                 CODD
-------------------- ----------
COMAN RADU           130000
1 record selected.
```

9) Să se selecteze cîmpurile MARCA, NUME, CODD, DEND, SALA+VENS pentru angajaţii care au salariul mai mare decît media salariilor realizate în depozitul din care fac ei parte.

```
SQL> SELECT MARCA,NUME,SALARIATI.CODD,
  2     DEND,SALA+VENS
  3  FROM SALARIATI,DEPOZITE
  4  WHERE SALA>
  5   (SELECT AVG(SALA) FROM SALARIATI
  6   WHERE SALARIATI.CODD=DEPOZITE.CODD)
  7  AND DEPOZITE.CODD=SALARIATI.CODD
  8  ORDER BY MARCA;
```


Scanned with OKEN Scanner

```
MARCA NUME        CODD   DEND      SALA+VENS
------------------------------------------------
1000  COMAN RADU  130000 AUTO      37500
2500  VLAD VASILE 160000 SPORT     38000
2550  FRINCU ION  160000 SPORT     73000
3755  DORU DAN    130000 AUTO      42000
4 records selected.
```

4.7. Utilizarea expresiilor, funcţiilor, variabilelor sistem şi pseudo-coloanelor în selectarea datelor

a) Expresii aritmetice

Expresiile aritmetice pot fi utilizate în comenzile de selectare a datelor. Ele se construiesc cu ajutorul operatorilor aritmetici, numelor de coloane şi constantelor.

Exemple:

1) Să se selecteze cîmpurile CODD, CODP, CODC, CANT*PRET din tabela COMENZI, pentru toate înregistrările al căror cod de depozit este 100000.

```
SQL> SELECT CODD,CODP,CODC,CANT*PRET
  2  FROM COMENZI,DEPOZITE
  3  WHERE CODD=100000;
CODD   CODP   CODC       CANT*PRET
-----------------------------------
100000 13333 121111        320000
100000 14444 121111         27000
100000 16666 121111        375000
100000 22222 121111        282000
```

2) Să se selecteze codurile produselor şi data pînă la care preţurile sînt admise pentru produsele din tabela COMENZI care au preţul negociat mai mare decît preţul mediu stabilit.

Scanned with OKEN Scanner

```
SQL> SELECT DISTINCT PRETURI.CODP,DATASF
  2  FROM PRETURI,COMENZI
  3  WHERE COMENZI.PRET>
  4     (PRETMIN+PRETMAX)/2;
CODP  DATASF
---------------------
14444 01-NOV-92
11111 30-AUG-92
12222 30-SEP-92
16666 01-NOV-92
13333 01-OCT-92
5 records selected.
```

3) Să se afişeze numele, marca, raportul VENS/SALA şi veniturile totale pentru şefii de depozite. Ordonarea datelor să fie făcută crescător după valorile raportului menţionat.

```
SQL> SELECT NUME,MARCA,VENS/SALA,
  2         VENS+SALA
  3  FROM SALARIATI
  4  WHERE FUNCT='SEF DEP'
  5  ORDER BY VENS/SALA DESC;
NUME        MARCA VENS/SALA          VENS+SALA
-------------------------------------------------------------
FRINCU ION    2550 1.0277777777777778 73000
DORU DAN      3755 .15068493150684932 42000
COMAN RADU    1000 .07142857142857143 37500
VLAD VASILE   2500  .0410958904109589  38000
```

4) Să se selecteze numele, codul depozitului, venitul total lunar şi anual prognozat din tabela SALARIATI, pentru vînzători. Ordonarea să fie făcută crescător după valorile cîmpului NUME.

```
SQL> SELECT NUME,CODD,SALA+VENS,
  2  (SALA+VENS)*12
  3  FROM SALARIATI WHERE FUNCT='VINZATO'
  4  ORDER BY NUME;
NUME        CODD   SALA+VENS  (SALA+VENS)*12
-------------------------------------------------------------
ALEXE IOAN    160000     26500          318000
```


Scanned with OKEN Scanner

```
AVRAM ION      100000    22200      266400
BARBU DAN      120000    22750      273000
CARMEN ANA     130000    26500      318000
DAN ION        160000    24850      298200
MANU DAN       160000    30000      360000
RADU IOANA     130000    23750      285000
SANDU ION      130000    25600      307200
VLAD ION       120000    28560      342720
9 records selected.
```

b) Funcţia NULL VALUE

O expresie aritmetică în care un operand este nul (valoarea NULL) are o valoare nulă (NULL).

De aceea, de multe ori, construirea corectă a expresiilor aritmetice presupune transformarea valorii NULL într-o altă valoare, eventual nulă faţă de operaţia aritmetică (cum ar fi, de exemplu, zero la adunare, unu la înmulţire etc.).

Se presupune că în cîmpul VENS s-au introdus şi valori NULL. Pentru astfel de valori, expresia SALA+VENS are valoarea NULL, indiferent ce valoare are SALA, ceea ce matematic nu este corect. De aceea, pentru exemplul anterior, se recomandă ca valoarea NULL a cîmpului VENS să fie transformată în zero, înaintea evaluării expresiei aritmetice.

Exemplu:

Să se selecteze coloanele NUME, MARCA, VENS, SALA+VENS din tabela SALARIATI, în condiţiile în care codul superiorului este 1000.

```
SQL> SELECT NUME,MARCA,VENS,SALA+VENS
  2  FROM SALARIATI
  3  WHERE CODS=1000;
NUME       MARCA          VENS  SALA+VENS
-------------------------------------------------------------
AVRAM ION  1111           1000  22200
```


Scanned with OKEN Scanner

```
BARBU DAN   1222              2000  22750
COMAN RADU 1000               2500  36000
VLAD ION     2650
```

Observaţie: Deoarece venitul suplimentar al salariatului VLAD ION este NULL, venitul lui total (SALA+VENS) a rezultat tot NULL. Utilizînd funcţia NVL, ca în exemplul de mai jos, venitul total este calculat corect.

```
SQL> SELECT NUME,
  2        SALA+NVL(VENS,0) VENIT_TOTAL
  3  FROM SALARIATI ORDER BY NUME;

NUME                                VENIT_TOTAL
----------                          ------------
ALEXE IOAN                             25700
AVRAM ION                              22200
BARBU DAN                              22750
CARMEN ANA                             26500
COMAN RADU                             37500
DAN ION                                24850
DORU DAN                               42000
FRINCU ION                             73000
MANU DAN                               30000
RADU IOANA                             23750
SANDU ION                              25600
VLAD ION                               28560
VLAD VASILE                            38000
13 records selected.
```

c) *Etichete de coloană*

În capul de tabel, rezultat în urma cererilor, apar numele coloanelor din baza de date. În locul acestora pot fi afişate *etichete de coloană* declarate în comenzile de definire a tabelelor.

Sintaxa de declarare este:

```
SELECT
   col1 nume-etich1,col2 nume-etich2,...
...;
```


Scanned with OKEN Scanner

Exemplu:

1) Să se selecteze denumirea depozitelor, codul acestora şi numărul de salariaţi ce îşi desfăşoară activitatea în cadrul lor, ordonate crescător după denumire.

```
SQL> SELECT DEND "CLIENTUL",CODD CODUL,
  2       NRSAL "NUMAR DE SALARIATI"
  3  FROM DEPOZITE
  4  ORDER BY DEND;
```

CLIENTUL	CODUL	NUMAR DE SALARIATI
ALIMENTAR	120000	11
AUTO	130000	5
MOBILA	100000	3
SPORT	160000	4
TEXTILE	140000	7

5 records selected.

d) Funcţii aritmetice

Funcţiile aritmetice sînt utilizate în cererile de selecţie. Sintaxa acestor funcţii este dată în *Anexa 3*.

Exemple:

1) Să se afişeze codul, denumirea şi cantitatea ridicată la pătrat a produselor cu unitatea de măsură 'BUC'.

```
SQL> SELECT CODP CODUL,DENP DENUMIRE,
  2  POWER(CANT,2) FROM PRODUSE
  3  WHERE UM='BUC' ORDER BY CODP;
```

CODUL	DENUMIRE	POWER(CANT,2)
14444	SCAUN D4	1296
11111	MESE 15/20	49
12222	FOTOLIU A3	144
13333	CANAPEA A7	36

Scanned with OKEN Scanner

Apavaloaie D. Daniel

15555 BIROU C6X4 81
5 records selected.

2) Să se selecteze în modul DISTINCT cîmpurile CODP, DENP, CODC,DENC şi să se calculeze [(CANT*PRET)/2], rotunjit la două zecimale.

```
SQL> SELECT DISTINCT
  2    COMENZI.CODP,DENP,
  3    COMENZI.CODC,DENC,
  4  ROUND(COMENZI.CANT*COMENZI.PRET/2,2)
  5      POSIBIL
  6  FROM COMENZI,CLIENTI,PRODUSE
  7  WHERE COMENZI.CODP=PRODUSE.CODP
  8  AND COMENZI.CODC=CLIENTI.CODC;

CODP  DENP       CODC   DENC   POSIBIL
-------------------------------------------------
13333 CANAPEA A7 121111 UNIT-2 160000
16666 PLACAJ 2/2  121111 UNIT-2 187500
14444 SCAUN D4    121111 UNIT-2  13500
3 records selected.
```

e) Funcţii caracter

Aceste funcţii operează asupra şirurilor de caractere. Ele se utilizează pentru transformarea literelor mari în mici sau invers, extragerea unui subşir dintr-un şir începînd cu o anumită poziţie, selectarea cuvintelor care se pronunţă asemănător cu un şir dat etc. Sintaxa lor este prezentată în Anexa 3.

Exemple:

1) Să se afişeze o situaţie finală prin care să fie redate cîmpurile NUME şi MARCA angajatului reunite într-un cîmp comun, denumit "INFORMATIE" şi cîmpul venituri totale anuale denumit VENIT_ANUAL. Selecţia este cerută pentru angajaţii cu codul superiorului egal cu 1000.

Scanned with OKEN Scanner

```
SQL> SELECT NUME || '-' || MARCA INFORMATIE
  2  (SALA+VENS)*12 VENIT_ANUAL
  3  FROM SALARIATI
  4  WHERE CODS=1000;
INFORMATIE          VENIT_ANUAL
--------------------------------------------------
AVRAM ION-1111       266400
BARBU DAN-1222       273000
COMAN RADU-1000      450000
VLAD ION-2650        342720
4 records selected.
```

2) Să se selecteze cîmpurile NUME şi FUNCT din tabela SALARIATI şi să se atribuie un cod de clasificare fiecărei funcţii. Codul este format dintr-o singură cifră şi are valorile: 1 pentru vînzător, 2 pentru director, 3 pentru restul funcţiilor.

```
SQL> SELECT NUME,FUNCT,
  2  DECODE(FUNCT,'VINZATO',
  3     1,'DIRECT',2,3)CLASIF_FUNCT
  4  FROM SALARIATI ORDER BY FUNCT,NUME;
NUME                                FUNCT CLASIF_FUNCT
-------------------------------------------------------------------------------
COMAN RADU                          SEF DEP          3
DORU DAN                            SEF DEP          3
FRINCU ION                          SEF DEP          3
VLAD VASILE                         SEF DEP          3
ALEXE IOAN                          VINZATO          1
AVRAM ION                           VINZATO          1
BARBU DAN                           VINZATO          1
CARMEN ANA                          VINZATO          1
DAN ION                             VINZATO          1
MANU DAN                            VINZATO          1
RADU IOANA                          VINZATO          1
SANDU ION                           VINZATO          1
VLAD ION                            VINZATO          1
13 records selected.
```

3) Să se afişeze salariul şi veniturile suplimentare ale tuturor angajaţilor, cu specificarea numelui numai pentru vînzători.Pentru

Scanned with OKEN Scanner

restul angajaţilor să se afişeze mesajul: ***Nu interesează***
Ordonarea să se facă după funcţie, în mod descrescător.

```
SQL> SELECT DECODE(FUNCT,'VINZATO',NUME,
  2  '***Nu interesează***') "NUMELE",
  3  SALA,NVL(VENS,0) FROM SALARIATI
  4  ORDER BY FUNCT DESC;
```

NUMELE	SALA	VENS
AVRAM ION	21200	1000
BARBU DAN	20750	2000
DAN ION	24500	350
MANU DAN	27500	2500
VLAD ION	25000	3500
SANDU ION	25600	0
CARMEN ANA	26500	0
RADU IOANA	20750	3000
ALEXE IOAN	25700	0
Nu interesează	35000	2500
Nu interesează	36500	1500
Nu interesează	36000	37000
Nu interesează	36500	5500

13 records selected.

4) Să se afişeze primele 5 caractere din nume, marca şi primul caracter din funcţie, pentru toţi angajaţii.

```
SQL> SELECT SUBSTR(NUME,1,5),MARCA,
  2  SUBSTR(FUNCT,1,1)
  3  FROM SALARIATI;
```

SUBST	MARCA	S
AVRAM	1111	V
BARBU	1222	V
COMAN	1000	S
DAN I	3500	V
VLAD	2500	S
MANU	3700	V
FRINC	2550	S
VLAD	2650	V

Scanned with OKEN Scanner

```
DORU            3755 S
SANDU           3760 V
CARME           3770 V
RADU            1680 V
ALEXE           3755 V
13 records selected.
```

5) Să se selecteze şi afişeze numele, funcţia şi salariul total pentru toţi angajaţii care au numele terminat cu litera N. Situaţia trebuie să aibă următoarea formă:

```
PERSOANA                        SALA+VENS
-------------------------------------------
NUME SALARIAT-funcţie     XXXXXXXXX
SQL> SELECT NUME || '-' ||
  2  LOWER(FUNCT) PERSOANA,
  3  SALA+VENS FROM SALARIATI
  4  WHERE UPPER(NUME) LIKE '%N'
PERSOANA                        SALA+VENS
-------------------------------------------
AVRAM ION-vinzato                 22200
BARBU DAN-vinzato                 22750
DAN ION-vinzato                   24850
MANU DAN-vinzato                  30000
FRINCU ION-sef dep                73000
VLAD ION-vinzato                  28560
DORU DAN-sef dep                  42000
SANDU ION-vinzato                 25600
ALEXE IOAN-vinzato
```

6) Să se selecteze cîmpurile CODC, DENC, STR şi NR din tabela CLIENTI, pentru clienţii cu cifra 1 pe prima poziţie a contului lor.

```
SQL> SELECT CODC,DENC,STR,NR
  2  FROM CLIENTI WHERE
  3  INSTR(CONT,'1',1)=1;
CODC DENC       STR       NR
-----------------------------------------
121111 UNIT-1   MOSILOR    104
211111 UNIT-2   DOROBANTI  18
```


Scanned with OKEN Scanner

7) Să se afişeaze numele şi marca acelor angajaţi al căror nume se pronunţă asemănător cu DORU DAN.

```
SQL> SELECT NUME,MARCA FROM SALARIATI
  2  WHERE SOUNDEX(NUME)=
  3        SOUNDEX('DORU DAN')
NUME                    MARCA
----------------------------------------
DORU DAN                 3755
DORU DANIEL              5565
2 records selected.
```

8) Să se selecteze din tabela SALARIATI coloanele NUME şi MARCA, pentru angajaţii a căror funcţie este asemănătoare fonetic cu şirul de caractere: 'VINZAT'.

```
SQL> SELECT NUME,MARCA FROM SALARIATI
  2  WHERE SOUNDEX(FUNCT)=
  3        SOUNDEX('VINZAT ')
NUME                    MARCA
----------------------------------------
AVRAM ION                1111
BARBU DAN                1222
DAN ION                  3500
MANU DAN                 3700

          .....

9 records selected.
```

f) Selecţia constantelor

În situaţiile finale, alături de valorile coloanelor sau de cele rezultate în urma aplicării funcţiilor şi operatorilor asupra lor, pot apare şi constante. Această selecţie se realizează cu ajutorul secvenţei:

```
SELECT col,...,'constantă',...,col,...
...;
```


Scanned with OKEN Scanner

Exemplu:
Să se slecteze constanta NUMELE SI CODUL CLIENTILOR: şi valorile cîmpurilor DENC, CODC pentru clienţii din strada Calea Moşilor şi cu un cod mai mare ca 100000.

```
SQL> SELECT 'NUMELE SI CODUL CLIENTILOR:',
  2         DENC,CODC
  3  FROM CLIENTI
  4  WHERE STR LIKE 'MOSILOR%' AND
  5        CODC>100000

NUMELE SI CODUL CLIENTILOR:  DENC    CODC
-------------------------------------------------------------
NUMELE SI CODUL CLIENTILOR: UNIT-2  121111
```

g) Funcţii aplicate asupra cîmpurilor de tip dată calendaristică

Pentru afişarea cîmpurilor de tip dată calendaristică sau pentru calcule în care sînt implicate aceste cîmpuri, există funcţii specifice, prezentate în Anexa 3.

Exemple:
1) Să se selecteze în modul DISTINCT codurile şi denumirile produselor, precum şi a datei pînă la care preţurile actuale sînt admise. Ordonarea să se facă crescător după valorile cîmpului CODP.

```
SQL> SELECT DISTINCT PRODUSE.CODP,
  2  PRODUSE.DENP,DATASF
  3  FROM PRETURI,PRODUSE
  4  WHERE PRODUSE.CODP=PRETURI.CODP
  5  ORDER BY PRETURI.CODP
CODP DENP        DATASF
-----------------------------------------
11111 MESE 15/20    30-AUG-92
```


Scanned with OKEN Scanner

12222 FOTOLIU A3 30-SEP-92
13333 CANAPEA A7 01-OCT-92
15555 BIROU C6X4 01-OCT-92
16666 PLACAJ 2/2 01-NOV-92
14444 SCAUN D4 01-NOV-92
6 records selected.

2) Să se selecteze cîmpurile CODP şi DATASF scriindu-se codul pe un rînd şi data pe următorul. Data va fi scrisă sub forma LL/ZZ-AA. Se vor folosi operatorul **TRUNC** care asigură trecerea la rîndul următor, funcţia **TO_CHAR(expr[,fmt])** care efectuează conversia cîmpului dată **expr** într-un şir de caractere în formatul specificat în [,fmt].De asemenea, prin comanda COLUMN să se definească un alt format de afişare a coloanei DATASF.

```
SQL> COLUMN DATASF FORMAT A21 TRUNC
SQL> SELECT CODP,TO_CHAR(DATASF,
  2  'MMDD-YY')DATASF FROM PRETURI;

CODP
    DATASF
---------- ---------------------
11111
    08/30-92
12222
    09/30-92
13333
    10/01-92
15555
    10/01-92
16666
    11/01-92
14444
    11/01-92
6 records selected.
```

3) Să se selecteze coloanele CODP, DATASF la produsele cu codul mai mare ca 13333, afişînd ziua şi luna în litere iar anul cu patru cifre.

```
SQL> COLUMN DATASF FORMAT A25
SQL> SELECT CODP,
```


Scanned with OKEN Scanner

```
  2  TO_CHAR(DATASF,
  3          'DAY MONTH YYYY')DATASF
  4  FROM PRETURI WHERE CODP>13333
CODP  DATASF
----- ---------------------------------------------
15555 THURSDAY OCTOBER   1992
16666 SUNDAY    NOVEMBER  1992
14444 SUNDAY    NOVEMBER  1992
```

4) Să se selecteze CODP, DATASF la produsele cu codul mai mare ca 13333, afişînd luna în litere şi pentru an ultimele două cifre.

```
SQL> SELECT CODP,
  2  TO_CHAR(DATASF,'DAY MONTH YY')DATASF
  3  FROM PRETURI WHERE CODP>13333
CODP DATASF
---- ------------------------------------------
15555 THURSDAY OCTOBER   92
16666 SUNDAY    NOVEMBER  92
14444 SUNDAY    NOVEMBER  92
```

5) Să se selecteze coloanele CODP, DATASF la produsele cu codul mai mare ca 13333, afişînd ziua în cifre, luna în litere şi anul cu patru cifre.

```
SQL> SELECT CODP,
  2  TO_CHAR(DATASF,
  3          'DD MONTH YYYY')DATASF
  4  FROM PRETURI WHERE CODP>13333
CODP DATASF
---- ---------------------------------
15555 01 OCTOBER   1992
16666 01 NOVEMBER  1992
14444 01 NOVEMBER  1992
```

6) Să se selecteze coloanele CODP, DATASF la produsele cu codul mai mare ca 13333, afişînd primele trei litere de la zi, luna în litere şi anul cu patru cifre.

```
SQL> SELECT CODP,
  2  TO_CHAR(DATASF,
  3          'DY MONTH YYYY')DATASF
```


Scanned with OKEN Scanner

```
  4  FROM PRETURI WHERE CODP>13333
CODP DATASF
------------------------------------
15555 THU OCTOBER   1992
16666 SUN NOVEMBER  1992
14444 SUN NOVEMBER  1992
```

7) Să se selecteze cîmpurile CODP, DATASF la produsele cu codul mai mare ca 13333, afişînd ziua în litere, luna în cifre şi anul cu patru cifre.

```
SQL> SELECT CODP,
  2  TO_CHAR(DATASF,'DAY MM YYYY')DATASF
  3  FROM PRETURI WHERE CODP>13333
CODP DATASF
---- ------------------------------------
15555 THURSDAY 10 1992
16666 SUNDAY    11 1992
14444 SUNDAY    11 1992
3 records selected.
```

8) Să se selecteze cîmpurile CODP, DATASF la produsele cu codul mai mare ca 13333, afişînd primele trei litere pentru zi şi lună şi ultimele două cifre de la an.

```
SQL> SELECT CODP,
  2  TO_CHAR(DATASF,'DY-MON-YY')DATASF
  3  FROM PRETURI WHERE CODP>13333
CODP DATASF
---- ------------------------
15555 THU-OCT-92
16666 SUN-NOV-92
14444 SUN-NOV-92
3 records selected.
```

9) Să se selecteze cîmpurile CODP, DATASF pentru produsele cu codul mai mare decît 13333, afişîndu-se primele trei litere de la zi, luna în litere şiultimele două cifre de la an.

```
SQL> SELECT CODP,
  2  TO_CHAR(DATASF,'DY-MONth-YY')DATASF
  3  FROM PRETURI WHERE CODP>13333
```


Scanned with OKEN Scanner

```
CODP        DATASF
-----------------------------
15555 THU-OCTOBER  -92
16666 SUN-NOVEMBER -92
14444 SUN-NOVEMBER -92
3 records selected.
```

10) Să se selecteze coloanele CODP, DATASF pentru toate produsele, afişînd ziua în cifre urmate de sufixul -th, primele trei litere ale lunii şi ultimele două cifre ale anului, despărţite prin liniuţă.

```
SQL> SELECT CODP,
  2  TO_CHAR(DATASF,
  3         'DDth-MON-YY')DATASF
  4  FROM PRETURI
CODP DATASF
-------------------------
11111 30TH-AUG-92
12222 30TH-SEP-92
13333 01ST-OCT-92
15555 01ST-OCT-92
16666 01ST-NOV-92
14444 01ST-NOV-92
6 records selected.
```

11) Să se selecteze din tabela PRETURI valorile cîmpurilor CODP şi DATASF. Data de sfîrşit (DATASF) să se prezinte însoţită de timpul intern, exprimat în diverse forme de afişare.

```
SQL> SELECT CODP,
  2  TO_CHAR(DATASF,
  3       'DDth-MON-YY HH:MIPM')DATASF
  4  FROM PRETURI
CODP DATASF
---------------------------------------
11111 30TH-AUG-92 12:00AM
12222 30TH-SEP-92 12:00AM
13333 01ST-OCT-92 12:00AM
15555 01ST-OCT-92 12:00AM
```


Scanned with OKEN Scanner

```
16666 01ST-NOV-92 12:00AM
14444 01ST-NOV-92 12:00AM
6 records selected.
```

sau :

```
SQL> SELECT CODP,
  2  TO_CHAR(DATASF,
  3     'DDth-MON-YY HH:MI')DATASF
  4  FROM PRETURI
CODP DATASF
------------------------------------
11111 30TH-AUG-92 12:00
12222 30TH-SEP-92 12:00
13333 01ST-OCT-92 12:00
15555 01ST-OCT-92 12:00
16666 01ST-NOV-92 12:00
14444 01ST-NOV-92 12:00
6 records selected.
```

12) Să se afişeze cîmpurile CODP, DATASF şi data de sfîrşit a valabilităţii unui preţ (considerată la o distanţă de 90 de zile faţă de DATASF).

```
SQL> SELECT CODP,DATASF,DATASF+90
  2  FROM PRETURI;
CODP  DATASF    DATASF+90
-----------------------------------------
11111 30-AUG-92 28-NOV-92
12222 30-SEP-92 29-DEC-92
13333 01-OCT-92 30-DEC-92
15555 01-OCT-92 30-DEC-92
16666 01-NOV-92 30-JAN-93
14444 01-NOV-92 30-JAN-93
```

13) Să se selecteze codul produsului, data maximă admisă de practicare a unui preţ şi data curentă pentru acele produse care îndeplinesc condiţia ca DATASF+10 să fie mai mare decît SYSDATE.

```
SQL> SELECT CODP,DATASF,
  2    SYSDATE DATA_CURENTA
```


Scanned with OKEN Scanner

```
 3  FROM PRETURI
 4  WHERE DATASF+10>SYSDATE
CODP DATASF        DATA_CURENTA
-------------------------------------------------------
12222 30-SEP-92      13-SEP-92
13333 01-OCT-92      13-SEP-92
15555 01-OCT-92      13-SEP-92
16666 01-NOV-92      13-SEP-92
14444 01-NOV-92      13-SEP-92
5 records selected.
```

14) Să se selecteze codul produsului, data de sfîrşit, data curentă, valorile expresiilor **TRUNC(DATASF+90)-TRUNC(SYSDATE)** şi **DATASF+90**. Selecţia să se realizeze pentru produsele cu codul mai mare ca 13333 şi data de sfîrşit plus 90 de zile mai mare decît data curentă.

```
SQL> SELECT CODP,DATASF,
2        SYSDATE DATA_C,
3  TRUNC(DATASF+90)-TRUNC(SYSDATE) DIF
4  DATASF+90 DATA_R
5  FROM PRETURI
6  WHERE DATASF+90>SYSDATE
7  AND CODP>13333;
CODP  DATASF   DATA_C     DIF   DATA_R
-------------------------------------------------------
15555 01-OCT-92 05-OCT-89    1182   13-DEC-92
16666 01-NOV-92 05-OCT-89    1213   30-JAN-93
14444 01-NOV-92 05-OCT-89    1213   30-JAN-93
3 records selected.
```

Observaţie: Operaţiile de calcul cu data calendaristică sînt posibile în cadrul unei comenzi de selecţie. Structura afişării cîmpurilor rezultate se poate stabili prin comanda COLUMN, specificată anterior.

Exemplu:

Să se selecteze şi afişeze coloanele CODP, DATASF, RDATE şi RSDATE (utilizînd funcţiile NEXT_DAY şi LAST_DAY).

Scanned with OKEN Scanner

```
SQL> COLUMN DATASF FORMAT A21
SQL> COLUMN RDATE FORMAT A21
SQL> COLUMN RSDATE FORMAT A20
SQL> SELECT CODP,
  2  TO_CHAR(DATASF,'DAY MON-YY')DATASF,
  3  TO_CHAR(NEXT_DAY(DATASF+90,'VINERI'),
  4      'DAY MON-YY') RDATE,
  5  TO_CHAR(LAST_DAY(DATASF+90),
  6      'DAY MON-YY')RSDATE
  7  FROM PRETURI;
```

CODP	DATASF	RSDATE	RDATE
11111	SUNDAY AUG-92	MONDAY NOV-92	VINERI DEC-92
12222	WEDNESDAY SEP-92	THURSDAY DEC-92	VINERI JAN-93
13333	THURSDAY OCT-92	THURSDAY DEC-92	VINERI JAN-93
15555	THURSDAY OCT-92	THURSDAY DEC-92	VINERI JAN-93
16666	SUNDAY NOV-92	SUNDAY JAN-93	VINERI FEB-93
14444	SUNDAY NOV-92	SUNDAY JAN-93	VINERI FEB-93

6 records selected.

O operaţie posibil de realizat este şi cea de a aduna algebric un număr de luni la o dată calendaristică (funcţia **ADD_MONTHS(d,n)**, care adună **n** luni la data **d**).

Exemple:

1) Să se afişeze codul produsului, data de sfîrşit şi a unui nou termen de valabilitate a unui preţ dat, calculat prin adăugarea a trei luni. Să se selecteze doar produsele al căror cod este mai mic decît 13333 şi data de sfîrşit este mai mare decît data curentă plus trei luni.

```
SQL> SELECT CODP,DATASF,
  2    ADD_MONTHS(DATASF,3)RDATE
```


Scanned with OKEN Scanner

```
 3  FROM PRETURI
 4  WHERE DATASF>SYSDATE+90
 5  AND CODP<13333;
CODP  DATASF    RDATE
-----------------------------------
11111 30-AUG-92 30-NOV-92
12222 30-SEP-92  31-DEC-92
```

2) Să se selecteze codul, denumirea produselor şi data de sfîrşit, pentru acele produse care îndeplinesc următoarele condiţii: data de sfîrşit este cuprinsă în intervalul: 30/Aug/92 +1/Oct/92 şi preţurile sînt mai mari ca 30.000 lei. Datele să se ordoneze crescător după codul produsului.

```
SQL> SELECT DISTINCT PRODUSE.CODP,DENP,
2     DATASF
3  FROM PRETURI,PRODUSE
4  WHERE
5       DATASF BETWEEN '30-AUG-92' AND
6       '01-OCT92' AND PRET>30000
7     AND PRODUSE.CODP=PRETURI.CODP
8  ORDER BY PRODUSE.CODP;
CODP DENP        DATASF
-----------------------------------------
12222 FOTOLIU A3   30-SEP-92
13333 CANAPEA A7   01-OCT-92
15555 BIROU C6X4   01-OCT-92
```

4.8. Utilizarea funcţiilor de grupare şi a clauzei GROUP BY în selectarea datelor

Rezultatele obţinute în urma selectărilor pot fi grupate cu ajutorul clauzei GROUP BY. Secvenţa utilizată pentru această operaţie este:

```
SELECT ...
    GROUP BY tabelă.col,tabelă.col,...
    HAVING condiţie ...
...;
```

Prin parcurgerea secvenţei se obţine cîte un rînd pentru înregis-

Scanned with OKEN Scanner

trările care au aceleaşi valori în coloanele specificate în clauza **GROUP BY**. Prezenţa clauzei **HAVING** determină obţinerea acelor grupuri care îndeplinesc condiţiile specificate.

Este de reţinut faptul că **GROUP BY** şi **HAVING** trebuie să fie declarate după clauzele **WHERE, CONNECT BY** şi **START WITH**, în cazul cînd acestea există în comandă.

Exemple:

1) Să se selecteze din tabela PRETURI valorile din coloana DATASF şi să se contorizeze numărul apariţiilor acestora.

```
SQL> SELECT TO_CHAR(DATASF,
  2    '"DATA-"DD MON YYYY')
  3    DATA_MAXIMA,COUNT(*) NUMAR_PRODUSE
  4 FROM PRETURI
  5 GROUP BY
  6    TO_CHAR(DATASF,
  7    '"DATA-"DD MON YYYY');
```

DATA_MAXIMA	NUMAR_PRODUSE
DATA-01 NOV 1992	2
DATA-01 OCT 1992	2
DATA-30 AUG 1992	1
DATA-30 SEP 1992	1

2) Să se selecteze şi afişeze valoarea medie zilnică a comenzilor ce trebuie onorate în perioada 01-30 Iulie 1992

```
SQL> SELECT AVG(CANT*PRET) FROM COMENZI
  2  WHERE DATAL > '01-JUL-92'
  3  AND DATAL < '30-JUL-92';
```

```
AVG(CANT*PRET)
--------------
         25100
```

3) Să se afişeze valoarea totală a salariilor şi veniturilor suplimentare pentru salariaţii cu funcţia 'VINZATO'.

```
SQL> SELECT SUM(SALA),SUM(VENS)
  2  FROM SALARIATI
```


Scanned with OKEN Scanner

```
  3  WHERE FUNCT='VINZATO';
       SUM(SALA)        SUM(VENS)
-----------------------------------------
          268560           14750
```

4) Să se afişeze media anuală a veniturilor totale (SALA+VENS) pentru salariaţii cu funcţia 'VINZATO'.

```
SQL> SELECT AVG(SALA+VENS)*12
  2  MEDIA_ANUALA
  3  FROM SALARIATI WHERE FUNCT='VINZATO';
MEDIA_ANUALA
------------------------
308147
```

5) Să se afişeze valoarea maximă şi minimă a salariului precum şi diferenţa dintre aceste două valori. Selecţia se face din tabela SALARIATI.

```
SQL> SELECT MAX(SALA),MIN(SALA),
  2  MAX(SALA)-MIN(SALA)
  3  FROM SALARIATI;
MAX(SALA) MIN(SALA) MAX(SALA)-MIN(SALA)
-----------------------------------------------------------
45000          24250       20750
```

6) Să se determine lungimea maximă a şirurilor de caractere din coloana DENP a tabelei PRODUSE.

```
SQL> SELECT MAX(LENGTH(DENP))
  2       FROM PRODUSE;
MAX(LENGTH(DENP))
-------------------------------
10
```

7) Să se selecteze coloanele NUME, FUNCT, SALA+VENS din tabela SALARIATI pentru angajaţii care au salariul egal cu salariul maxim.

```
SQL> SELECT NUME,FUNCT,SALA+VENS
  2  FROM SALARIATI
  3  WHERE SALA=
```


Scanned with OKEN Scanner

```
  4    (SELECT MAX(SALA) FROM SALARIATI);
NUME            FUNCT          SALA+VENS
--------------------------------------------------
ION ION         DIRECTO              85000
```

8) Să se selecteze coloanele DENP, CODP, CODD din tabela PRODUSE pentru care stocul existent este mai mare sau egal cu cantitatea comandată.

```
SQL> SELECT DENP,CODP,CODD
  2  FROM PRODUSE
  3  WHERE STOC>=
  4    (SELECT SUM(CANT)
  5    FROM COMENZI);
DENP          CODP     CODD
------------------------------ --------
PLACAJ 2/2    166666   100000
SCAUN D4      144444   100000
CANAPEA A7    133333 100000
```

9) Să se afişeze numărul de valori nenule înregistrate în coloana VENS din tabela SALARIATI.

```
SQL> SELECT COUNT(VENS)
  2  FROM SALARIATI;
     COUNT(VENS)
-----------------------
              11
```

10) Să se selecteze din tabela SALARIATI coloanele NUME, FUNCT, SALA+VENS pentru şefii de depozite care au salariul mai mare sau egal cu jumătate din salariul maxim.

```
SQL> SELECT NUME,FUNCT,SALA+NVL(VENS,0)
  2  FROM SALARIATI WHERE SALA>=
  3  (SELECT MAX(SALA)/2 FROM SALARIATI)
  4  AND FUNCT='SEF DEP';
NUME            FUNCT     SALA+NVL(VENS,0)
-----------------------------------------------------
COMAN RADU      SEF DEP              37500
```


Scanned with OKEN Scanner

```
VLAD VASILE.      SEF DEP          38000
FRINCU ION        SEF DEP          73000
DORU DAN          SEF DEP          42000
PAUL STEFAN       SEF DEP          40600
```

11) Să se afişeze numărul de produse distincte care au ca unitate de măsură 'BUC'.

```
SQL> SELECT COUNT(DISTINCT CODP) NUMAR
  2  FROM PRODUSE
  3  WHERE UM='BUC';
NUMAR
---------
      5
```

12) Să se afişeze numărul de subordonaţi ai salariatului cu marca 2500.

```
SQL> SELECT COUNT(*) NR_SUBORDONATI
  2  FROM SALARIATI WHERE CODS=2500;
NR_SUBORDONATI
--------------------------
             3
```

13) Să se afişeze valoarea medie a salariilor şi valoare medie a veniturilor suplimentare pentru fiecare depozit fie utilizînd comanda SELECT, de mai multe ori, fie clauza *GROUP BY*.

```
SQL> SELECT AVG(SALA),AVG(VENS)
  2  FROM SALARIATI
  3  WHERE CODD=100000;
AVG(SALA)  AVG(VENS)
-------------------------------
33100      20500
SQL> SELECT AVG(SALA),AVG(VENS)
  2  FROM SALARIATI
  3  WHERE CODD=120000;
AVG(SALA) AVG(VENS)
-------------------------------
 24202.5      1975
```


Scanned with OKEN Scanner

```
SQL> SELECT AVG(SALA),AVG(VENS)
  2  FROM SALARIATI
  3  WHERE CODD=130000;
AVG(SALA) AVG(VENS)
------------------------------
    28870      2750
SQL> SELECT AVG(SALA),AVG(VENS)
  2  FROM SALARIATI
  3  WHERE CODD=160000;
AVG(SALA) AVG(VENS)
------------------------------
  30866.7      9390
SQL> SELECT CODD,AVG(SALA),AVG(VENS)
  2  FROM SALARIATI
  3  GROUP BY CODD;
   CODD AVG(SALA) AVG(VENS)
---------- ---------- ----------
 100000      33100      20500
 120000    24202.5       1975
 130000      28870       2750
 160000    30866.7       9390
```

14) Să se calculeze media salariului pentru fiecare grup de angajaţi care au acelaşi superior.

```
SQL> SELECT CODS,AVG(SALA)
  2  FROM SALARIATI
  3  GROUP BY CODS;
          CODS         AVG(SALA)
---------- ----------
          1000         22336,7
          2500         27083,3
          2550         25500
          3755         26600
          4000         26050
          7000         35750
          8000          4500
```


Scanned with OKEN Scanner

15) Să se calculeze media salariului anual prognozat pentru salariaţii care au acelaşi superior şi nu au funcţia de şef depozit.

```
SQL> SELECT CODS,AVG(SALA)*12
  2  FROM SALARIATI
  3  WHERE FUNCT NOT IN('SEF DEP')
  4  GROUP BY CODS;
CODS       AVG(SALA)*12
-----------------------------------
1000  268040
2500  271500
2550  306000
3755  319200
4000  312600
8000  540000
```

16) Să se calculeze şi afişeze valoarea medie a comenzilor pentru fiecare depozit.

```
SQL> SELECT AVG(CANT*PRET),
  2         DEPOZITE.CODD
  3  FROM COMENZI,DEPOZITE
  4  WHERE COMENZI.CODD=DEPOZITE.CODD
  5  GROUP BY DEPOZITE.CODD
   AVG(CANT*PRET)              CODD
-----------------------------------------------
         251000                 100000
```

17) Să se calculeze şi afişeze valoarea medie a comenzilor pe produse.

```
SQL> SELECT AVG(CANT*PRET),PRODUSE.CODP
  2  FROM PRODUSE,COMENZI
  3  WHERE COMENZI.CODP=PRODUSE.CODP
  4  GROUP BY PRODUSE.CODP;
AVG(CANT*PRET)    CODP
-----------------------------------
       320000       133333
        27000       144444
       375000       166666
```


Scanned with OKEN Scanner

18) Să se determine media salariului anual pentru fiecare funcţie dacă există mai mult de doi salariaţi angajaţi pe aceeaşi funcţie. Se vor afişa funcţia, numărul de salariaţi cu funcţia respectivă şi media calculată.

```
SQL> SELECT FUNCT,COUNT(*),AVG(SALA)*12
  2  FROM SALARIATI
  3  GROUP BY FUNCT
  4  HAVING COUNT(*)>2;

FUNCT          COUNT(*)      AVG(SALA)*12
-------------------------------------------
SEF DEP             5            429600
VINZATO            11            292975
2 records selected.
```

19) Să se selecteze codurile superiorilor (CODS) care au doi sau mai mulţi subordonaţi.

```
SQL> SELECT CODS FROM SALARIATI
  2  WHERE FUNCT='VINZATO'
  3  GROUP BY CODS
  4  HAVING COUNT(*)>=2;

      CODS
----------
      1000
      2500
      2550
      3755
      4000
5 records selected.
```

20) Să se selecteze codurile superiorilor care au media veniturilor suplimentare ale subordonaţilor mai mare decît 10% din salariul mediu. Se vor afişa codurile superiorilor şi media veniturilor totale anuale ale subordonaţilor.

```
SQL> SELECT CODS,AVG(SALA+NVL(VENS,0))*12
  2  FROM SALARIATI
  3  GROUP BY CODS
  4  HAVING AVG(VENS)>AVG(SALA)*0.10;
```


Scanned with OKEN Scanner

```
CODS  AVG(SALA+NVL(VENS,0))*12
-----------------------------------------------------
2500                 486400
7000                 474300
8000                 1020000
```

21) Să se selecteze funcţiile pentru care salariile medii sînt mai mari decît salariul mediu al unui vînzător. Se vor afişa funcţiile şi salariile medii pentru fiecare funcţie.

```
SQL> SELECT FUNCT,AVG(SALA)
  2 FROM SALARIATI GROUP BY FUNCT
  3 HAVING AVG(SALA)>
  4 (SELECT AVG(SALA) FROM SALARIATI
  5    WHERE FUNCT='VINZATO');
FUNCT          AVG(SALA)
-----------------------------------
DIRECTO          45000
SEF DEP          35800
```

22) Să se afişeze numărul salariaţilor din depozitul cu codul 1000 şi să se determine cîţi dintre ei au venituri suplimentare.

```
SQL> SELECT COUNT(*) NR_SALA,
  2 COUNT(VENS) VENIT_SUPLIMENTAR
  3 FROM SALARIATI WHERE CODS=1000
      NR_SALA  VENIT_SUPLIMENTAR
-----------------------------------------------------
         4              3
```

23) Să se determine numărul salariaţilor din depozitul cu codul 1000 precum şi suma şi media veniturilor lor suplimentare.

```
SQL> SELECT SUM(NVL(VENS,0)) SUMA,
  2       COUNT(NVL(VENS,0)) NUMAR,
  3       AVG(NVL(VENS,0)) MEDIA
  4 FROM SALARIATI WHERE CODS=1000
SUMA       NUMAR          MEDIA
-----------------------------------------------
9000       4              2250
```


Scanned with OKEN Scanner

4.9. Operaţii pe tabele structurate arborescent

Limbajul SQL*PLUS permite explorarea structurilor arborescente existente în baza de date. Operaţia se realizează cu ajutorul clauzelor START WITH şi CONNECT BY din comanda SELECT:

```
SELECT ...
   FROM ...
   CONNECT BY [PRIOR] col1= [PRIOR] col2
   START WITH col=valoare
...;
```

Pentru exemplificare, se consideră structura arborescentă din figura 4.1., dată de modul de subordonare a salariaţilor.

Exemple:

1) Să se afişeze cîmpurile NUME, FUNCT, CODS, MARCA din tabela SALARIATI. Datele să fie ordonate crescător după codul superiorului.

```
SQL> SELECT NUME,FUNCT,CODS,MARCA
  2  FROM SALARIATI
  3  ORDER BY CODS;
```

NUME	FUNCT	CODS	MARCA
AVRAM ION	VINZATO	1000	1111
BARBU DAN	VINZATO	1000	1222
COMAN RADU	SEF DEP	1000	1000
VLAD ION	VINZATO	1000	2650
DAN ION	VINZATO	2500	3500
VLAD VASILE	SEF DEP	2500	2500
MANU DAN	VINZATO	2500	3700
FRINCU ION	SEF DEP	2500	2550
RADU ION	VINZATO	2500	1680
ALEXE IOAN	VINZATO	2500	3755
AILENEI FLORIN	VINZATO	2550	2553
ALEXE IOAN	VINZATO	3775	3759
DORU DAN	SEF DEP	4000	3755

Scanned with OKEN Scanner

Figura 4.1

Scanned with OKEN Scanner

SANDU ION	VINZATO	4000	3760
CARMEN ANA	VINZATO	4000	3770
PAUL STEFAN	SEF DEP	7000	4000
ION ION	DIRECTO	8000	7000

17 records selected.

2) Să se afişeze cîmpurile MARCA, NUME, FUNCT, CODS, CODD din tabela SALARIATI, ordonate crescător după MARCA.

```
SQL> SELECT MARCA,NUME,FUNCT,CODS,CODD
  2  FROM SALARIATI
  3  ORDER BY MARCA;
```

MARCA	NUME	FUNCT	CODS	CODD
1000	COMAN RADU	SEF DEP	7000	130000
1111	AVRAM ION	VINZATO	1000	100000
1222	BARBU DAN	VINZATO	1000	120000
1680	RADU ION	VINZATO	2500	130000
2500	VLAD VASILE	SEF DEP	7000	160000
2550	FRINCU ION	SEF DEP	2500	160000
2553	AILENEI FLORIN	VINZATO	2550	120000
2554	DARIAN GEO	VINZATO	2550	120000
2650	VLAD ION	VINZATO	1000	120000
3500	DAN ION	VINZATO	2500	160000
3700	MANU DAN	VINZATO	3755	160000
3755	DORU DAN	SEF DEP	7000	130000
3759	ALEXE IOAN	VINZATO	3755	160000
3760	SANDU ION	VINZATO	4000	130000
3770	CARMEN ANA	VINZATO	4000	130000
4000	PAUL STEFAN	SEF DEP	7000	160000
7000	ION ION	DIRECTO	7000	100000

17 records selected.

3) Să se selecteze numele, marca, codul superiorului, codul depozitului pentru subordonaţii direcţi şi indirecţi ai salariatului cu numele DORU DAN.

```
SQL> SELECT NUME,MARCA,CODS,CODD
  2  FROM SALARIATI
```


Scanned with OKEN Scanner

```
3 CONNECT BY PRIOR MARCA=CODS
4 START WITH NUME='DORU DAN';
```

NUME	MARCA	CODS	CODD
DORU DAN	3755	7000	130000
MANU DAN	3700	3755	160000
ALEXE IOAN	3759	3755	160000

3 records selected.

4) Să se afişeze MARCA, NUME, FUNCT, CODS şi CODD ordonate crescător după marcă şi cod superior pentru subordonaţii direcţi şi indirecţi ai salariatului cu numele VLAD VASILE.

```
SQL> SELECT MARCA,NUME,FUNCT,CODS,CODD
2 FROM SALARIATI
3 CONNECT BY PRIOR MARCA=CODS
4 START WITH NUME='VLAD VASILE'
5 ORDER BY MARCA,CODS
```

MARCA	NUME	FUNCT	CODS	CODD
1680	RADU ION	VINZATO	2500	130000
2500	VLAD VASILE	SEF DEP	7000	160000
2550	FRINCU ION	SEF DEP	2500	160000
2553	AILENEI FLORIN	VINZATO	2550	120000
2554	DARIAN GEO	VINZATO	2550	120000
3500	DAN ION	VINZATO	2500	160000

5) Să se afişeze cîmpurile NUME şi MARCA pentru subordonaţii direcţi şi indirecţi ai salariatului ION ION precum şi nivelul lor de subordonare.

```
SQL> SELECT LEVEL,NUME,MARCA
2 FROM SALARIATI
3 CONNECT BY PRIOR MARCA=CODS
4 START WITH NUME='ION ION';
```

LEVEL	NUME	MARCA
1	ION ION	7000
2	COMAN RADU	1000

Scanned with OKEN Scanner

3 AVRAM ION	1111
3 BARBU DAN	1222
3 VLAD ION	2650
2 VLAD VASILE	2500
3 FRINCU ION	2550
4 AILENEI FLORIN	2553
4 DARIAN GEO	2554
3 RADU ION	1680
3 DAN ION	3500
2 DORU DAN	3755
3 MANU DAN	3700
3 ALEXE IOAN	3759
2 PAUL STEFAN	4000
3 SANDU ION	3760
3 CARMEN ANA	3770

6) Să se selecteze cîmpurile LEVEL, NUME şi MARCA aparţinînd subordonaţilor lui ION ION. Nivelul de subordonare va fi pus în evidenţă şi prin afişarea deplasată a numelor salariaţilor subordonaţi faţă de numele superiorului corespunzător.

```
SQL> COLUMN ORG_CHART FORMAT A21
SQL> SELECT LEVEL,
 2  LPAD(' ',LEVEL*2)||NUME NUME_SI_PRENUME,
 3  MARCA
 4  FROM SALARIATI
 5  CONNECT BY PRIOR MARCA=CODS
 6  START WITH NUME='ION ION';
```

LEVEL	NUME_SI_PRENUME	MARCA
1	ION ION	7000
2	COMAN RADU	1000
3	AVRAM ION	1111
3	BARBU DAN	1222
3	VLAD ION	2650
2	VLAD VASILE	2500
3	FRINCU ION	2550

Scanned with OKEN Scanner

```
 4        AILENEI FLORIN                2553
 4        DARIAN GEO                    2554
 3      RADU ION                        1680
 3      DAN ION                         3500
 2    DORU DAN                          3755
 3      MANU DAN                        3700
 3      ALEXE IOAN                      3759
 2    PAUL STEFAN                       4000
 3      SANDU ION                       3760
 3      CARMEN ANA                      3770
17 records selected.
```

7) Să se afişeze cîmpurile LEVEL, NUME, MARCA aparţinînd subordonaţilor salariaţilor PAUL STEFAN şi COMAN RADU, punînd în evidenţă, prin scriere decalată, modul de subordonare.

```
SQL> SELECT LEVEL,
  2      LPAD(' ',LEVEL*2)||NUME
  3            NUME_SI_PRENUME,MARCA
  4  FROM SALARIATI
  5  CONNECT BY PRIOR MARCA=CODS
  6  START WITH NUME='PAUL STEFAN'
  7  OR NUME='COMAN RADU'
  LEVEL NUME_SI_PRENUME                MARCA
---------------------------------------------
     1  COMAN RADU                      1000
     2    AVRAM ION                     1111
     2    BARBU DAN                     1222
     2    VLAD ION                      2650
     1  PAUL STEFAN                     4000
     2    SANDU ION                     3760
     2    CARMEN ANA                    3770
7 records selected.
```

8) Să se afişeze cîmpurile LEVEL, NUME şi MARCA, în structură arborescentă, aparţinînd salariaţilor din subordinea celor cu aceeaşi funcţie cu a salariatului COMAN RADU.

Scanned with OKEN Scanner

```
SQL> SELECT LEVEL,
  2       LPAD(' ',LEVEL*2)||NUME
  3          NUME_SI_PRENUME,
  4       MARCA
  5  FROM SALARIATI
  6  CONNECT BY PRIOR MARCA=CODS
  7  START WITH  FUNCT IN
  8     (SELECT FUNCT
  9     FROM SALARIATI
 10     WHERE NUME='COMAN RADU');
```

LEVEL	NUME_SI_PRENUME	MARCA
2	COMAN RADU	1000
3	AVRAM ION	1111
3	BARBU DAN	1222
3	VLAD ION	2650
2	VLAD VASILE	2500
3	FRINCU ION	2550
4	AILENEI FLORIN	2553
4	DARIAN GEO	2554
3	RADU ION	1680
3	DAN ION	3500
2	DORU DAN	3755
3	MANU DAN	3700
3	ALEXE IOAN	3759
2	PAUL STEFAN	4000
3	SANDU ION	3760
3	CARMEN ANA	3770

9) Să se selecteze cîmpurile LEVEL, NUME, MARCA, FUNCT, CODD aparţinînd subordonaţilor angajaţilor care lucrează în acelaşi depozit cu RADU ION.

```
SQL> SELECT LEVEL,
  2       LPAD(' ',LEVEL*2)||NUME
  3       NUME_SI_PRENUME,
  4       MARCA,FUNCT,CODD
  5  FROM SALARIATI.
```


Scanned with OKEN Scanner

```
 6  CONNECT BY PRIOR MARCA=CODS
 7  START WITH CODD IN
 8       (SELECT CODD
 9       FROM SALARIATI
10       WHERE NUME='RADU ION')
```

LEVEL	NUME_SI_PRENUME	MARCA	FUNCT	CODD
1	RADU ION	1680	VINZATO	130000
1	COMAN RADU	1000	SEF DEP	130000
2	AVRAM ION	1111	VINZATO	100000
2	BARBU DAN	1222	VINZATO	120000
2	VLAD ION	2650	VINZATO	120000
1	DORU DAN	3755	SEF DEP	130000
2	MANU DAN	3700	VINZATO	160000
2	ALEXE IOAN	3759	VINZATO	160000
1	CARMEN ANA	3770	VINZATO	130000
1	SANDU ION	3760	VINZATO	130000

10 records selected.

10) Să se selecteze cîmpurile NUME, MARCA, FUNCT, CODD aparţinînd superiorilor salariatului VLAD ION.

```
SQL> SELECT NUME,MARCA,CODS,FUNCT,CODD
  2  FROM SALARIATI
  3  CONNECT BY MARCA=PRIOR CODS
  4  START WITH NUME='VLAD ION';
```

NUME	MARCA	CODS	FUNCT	CODD
VLAD ION	2650	1000	VINZATO	120000
COMAN RADU	1000	7000	SEF DEP	130000
ION ION	7000	8000	DIRECTO	100000

3 records selected.

11) Să se afişeze cîmpurile NUME, MARCA, FUNCT aparţinînd subordonaţilor salariatului VLAD VASILE, mai puţin datele salariatului AILENEI FLORIN.

```
SQL> SELECT NUME,MARCA,FUNCT
  2  FROM SALARIATI
```


Scanned with OKEN Scanner

```
3 WHERE NUME !='AILENEI FLORIN'
4 CONNECT BY PRIOR MARCA=CODS
5 START WITH NUME='VLAD VASILE';
```

NUME	MARCA	FUNCT
VLAD VASILE	2500	SEF DEP
FRINCU ION	2550	SEF DEP
DARIAN GEO	2554	VINZATO
RADU ION	1680	VINZATO
DAN ION	3500	VINZATO

12) Să se afişeze o serie de date despre subordonaţii salariatului ION ION, mai puţin datele despre VLAD VASILE şi despre salariaţii din subordinea lui.

```
SQL> SELECT MARCA,LEVEL,NUME,FUNCT,CODS
2 FROM SALARIATI
3 CONNECT BY PRIOR MARCA=CODS
4 AND NUME != 'VLAD VASILE'
5 START WITH NUME='ION ION';
```

MARCA	LEVEL	NUME	FUNCT	CODS
7000	1	ION ION	DIRECTO	8000
1000	2	COMAN RADU	SEF DEP	7000
1111	3	AVRAM ION	VINZATO	1000
1222	3	BARBU DAN	VINZATO	1000
2650	3	VLAD ION	VINZATO	1000
3755	2	DORU DAN	SEF DEP	7000
3700	3	MANU DAN	VINZATO	3755
3759	3	ALEXE IOAN	VINZATO	3755
4000	2	PAUL STEFAN	SEF DEP	7000
3760	3	SANDU ION	VINZATO	4000
3770	3	CARMEN ANA	VINZATO	4000

11 records selected.

13) Să se afişeze o serie de date ale salariaţilor subordonaţi lui ION ION, mai puţin datele salariaţilor COMAN RADU şi VLAD VASILE precum şi ale subordonaţilor lui VLAD VASILE.

Scanned with OKEN Scanner

```
SQL> SELECT MARCA,NUME,LEVEL,FUNCT,
  2       CODS,CODD
  3  FROM SALARIATI
  4  WHERE NUME != 'COMAN RADU'
  5  CONNECT BY PRIOR MARCA=CODS
  6  AND NUME != 'VLAD VASILE'
  7  START WITH NUME='ION ION'
```

MARCA	NUME	LEVEL	FUNCT	CODS	CODD
7000	ION ION	1	DIRECTO	8000	100000
1111	AVRAM ION	3	VINZATO	1000	100000
1222	BARBU DAN	3	VINZATO	1000	120000
2650	VLAD ION	3	VINZATO	1000	120000
3755	DORU DAN	2	SEF DEP	1000	130000
3700	MANU DAN	3	VINZATO	3755	160000
3759	ALEXE IOAN	3	VINZATO	3755	160000
4000	PAUL STEFAN	2	SEF DEP	7000	160000
3760	SANDU ION	3	VINZATO	4000	130000
3770	CARMEN ANA	3	VINZATO	4000	130000

10 records selected.

14) Să se afişeze datele salariaţilor subordonaţi direct salariatului ION ION.

```
SQL> SELECT MARCA,NUME,LEVEL,
  2       FUNCT,CODS,CODD
  3  FROM SALARIATI
  4  WHERE  LEVEL=2
  5  CONNECT BY PRIOR MARCA=CODS
  6  START WITH NUME='ION ION'
```

MARCA	NUME	LEVEL	FUNCT	CODS	CODD
1000	COMAN RADU	2	SEF DEP	7000	130000
2500	VLAD VASILE	2	SEF DEP	7000	160000
3755	DORU DAN	2	SEF DEP	7000	130000
4000	PAUL STEFAN	2	SEF DEP	7000	160000

4 records selected.

Scanned with OKEN Scanner

15) Să se calculeze şi afişeze media salariului pe fiecare nivel pentru subordonaţii lui ION ION.

```
SQL> SELECT LEVEL,AVG(SALA)
  2  FROM SALARIATI
  3  CONNECT BY PRIOR MARCA=CODS
  4  START WITH NUME='ION ION'
  5  GROUP BY LEVEL
  6  ORDER BY LEVEL;

 LEVEL AVG(SALA)
------------------------
     1      45000
     2      35750
     3      25356
     4      25500
```


Scanned with OKEN Scanner

CAPITOLUL 5
ACTUALIZAREA DATELOR

Actualizarea datelor se referă la adăugarea unor noi rînduri într-o tabelă (cu comanda **INSERT**), la modificarea valorilor uneia sau mai multor valori dintr-un rînd (cu comanda **UPDATE**) şi la ştergerea unui rînd dintr-o tabelă (cu comanda **DELETE**).

În funcţie de momentul în care se doreşte realizarea modificărilor asupra bazei de date, utilizatorul poate folosi una din următoarele comenzi:

a) **SET AUTOCOMMIT IMM** - schimbările se efectuează imediat;

b) **SET AUTOCOMMIT OFF** - schimbările sînt păstrate într-un buffer pînă la execuţia uneia din comenzile: COMMIT WORK care are rolul de a permanentiza schimbările efectuate; ROLLBACK WORK care determină renunţarea la schimbările realizate.

5.1. Adăugarea rîndurilor într-o tabelă

În vederea adăugării unor rînduri noi într-o tabelă sau într-o viziune se utilizează comanda:

```
INSERT INTO nume-tabelă
      [(nume-col1,nume-col2,...)]
      {VALUES (valoare1,valoare2,...) | cerere};
```

Pentru nume-col1,nume-col2... precizate în paranteze vor fi furnizate valorile corespunzătoare, iar coloanelor nespecificate le sînt ataşate valori nule. Coloanele pot fi precizate în orice ordine, însă trebuie asigurată corespondenţa între numele coloanelor şi valorile furnizate.

În cazul în care anumite coloane nu sînt specificate explicit se impune ca ordinea în care apar valorile în comanda **INSERT** să coincidă cu cea în care coloanele au fost definite la crearea tabelei.

Dacă nu se mai cunoaşte ordinea de declarare a coloanelor se foloseşte comanda DESCRIBE care va afişa lista coloanelor definite pentru tabela respectivă, tipul şi lungimea lor.

Prin forma **INSERT...VALUES** se introduce în tabelă un singur

Scanned with OKEN Scanner

rînd. Cu ajutorul valorii **NULL** se pot introduce valori nule. Pentru a furniza valori pentru o coloană de tip dată calendaristică se poate folosi funcţia **TO_DATE** sau cuvîntul cheie **SYSDATE**.

Funcţia **TO_DATE** permite furnizarea valorilor într-un format diferit de cel standard.

Specificarea cererii din comanda **INSERT** determină copierea unor date dintr-o tabelă în alta pe atîtea rînduri cîte au rezultat din cererea SQL.

Exemple:

1) Să se adauge un nou rînd în tabela PRODUSE.

```
SQL> INSERT INTO PRODUSE VALUES
  1 (100000,11111,'MESE 15/20',7,'27-JUN-92','BUC')
1 record created.
```

2) Să se creeze o nouă tabelă, EXEMPLU, cu un singur cîmp numeric, identic cu coloana CODP a tabelei PRODUSE. Să se introducă în această nouă tabelă codul produselor din tabela PRODUSE.

```
SQL> CREATE TABLE EXEMPLU
  2  (CODP NUMBER(6) NOT NULL);
Table created.
SQL> INSERT INTO EXEMPLU
  2  SELECT CODP FROM PRODUSE;
5 records created.
```

5.2. Ştergerea rîndurilor dintr-o tabelă

Ştergerea unor rînduri dintr-o tabelă se realizează cu următoarea comandă:

```
DELETE FROM nume-tabelă [WHERE condiţie];
```

Folosirea clauzei **WHERE** determină ştergerea acelor rînduri care îndeplinesc condiţia impusă. În această clauză pot fi folosite şi

Scanned with OKEN Scanner

subcereri.

Dacă nu este specificată nici o condiţie, se şterg toate rîndurile tabelei.

Ştergerile accidentale pot fi omise, restaurîndu-se valorile iniţiale prin comanda **AUTOCOMMIT OFF**.

Exemple:

1) Să se şteargă datele din tabela DEPOZITE.

```
SQL> DELETE FROM DEPOZITE;
5 records deleted.
```

2) Să se şteargă datele pentru depozitele care au codul mai mare sau egal cu 100000.

```
SQL> DELETE FROM DEPOZITE WHERE
  2  CODD>=140000;
2 records deleted.
```

3) Să se scrie comanda pentru ştergerea datelor despre salariatul VLAD VASILE. Ştergerile să nu fie efectuate imediat ci ulterior.

```
SQL> SET AUTOCOMMIT OFF                    (Ştergerile nu
SQL> DELETE FROM SALARIATI                 se fac imediat)
  2   WHERE NUME='VLAD VASILE';
1 record deleted.
SQL> COMMIT WORK                           (Ştergerile se
commit complete                            realizează fizic)
```

4) Să se şteargă datele salariaţilor care au aceeaşi funcţie cu a lui PAUL STEFAN. Ştergerile să nu fie realizate imediat. Ulterior să se renunţe la aceste ştergeri.

```
SQL> SET AUTOCOMMIT OFF                    (Ştergerile nu
SQL> DELETE FROM SALARIATI                 se fac imediat)
  2   WHERE FUNCT IN
  3   (SELECT FUNCT
  4   FROM SALARIATI
  5   WHERE NUME='PAUL STEFAN');
5 records deleted.
```


Scanned with OKEN Scanner

```
SQL> ROLLBACK WORK                    (Se renunţă
rollback complete                      la ştergeri)
```

5.3. Modificarea datelor dintr-o tabelă

În scopul modificării datelor dintr-o tabelă se utilizează una din formele sintactice ale comenzii UPDATE:

```
UPDATE nume-tabelă [sinonim]
  SET nume-col=expresie,nume-col=expresie,...
  [WHERE condiţie];

UPDATE nume-tabelă [sinonim]
  SET (nume-col,nume-col,...)=(subcerere)
  [WHERE condiţie];
```

Se observă că sînt două posibilităţi de modificare. Una constă în furnizarea în mod explicit a fiecărei valori pentru cîmpurile care trebuie modificate iar cealaltă posibilitate constă în obţinerea valorilor în urma unei cereri SQL.

Dacă nu este specificată clauza WHERE se vor modifica toate rîndurile tabelei.

(nume-col,nume-col,...)=(subcerere) impune ca cererea să conţină pentru fiecare rînd un număr de valori corespunzător numărului de coloane din paranteza care precede caracterul =.

În cazul în care se modifică o singură coloană, parantezele pot fi omise.

Exemple:

1) Să se modifice cîmpul NRSAL din tabela SALARIATI, pentru depozitul cu codul 130000, atribuindu-i valoarea 11.

Scanned with OKEN Scanner

```
SQL> UPDATE DEPOZITE
  2  SET NRSAL=11
  3  WHERE CODD=130000;
1 record updated.
```

2) Să se modifice data de livrare, cantitatea solicitată şi preţul de livrare pentru produsul cu codul 13333 din comanda cu numărul 211111.

```
SQL> UPDATE COMENZI
  2  DATAL=SYSDATE,CANT=50, PRET=42000
  3  WHERE CODP=13333 AND
  4  NRCOM=211111;
1 record updated.
```

3) Să se modifice data de livrare cu data actuală pentru toate produsele cu codul egal cu 13333, din toate comenzile.

```
SQL> UPDATE COMENZI
  2  SET DALAL=SYSDATE
  3  WHERE CODP=13333;
1 record updated.
```

4) Să se mărească salariul cu 15% pentru salariaţii care au o funcţie identică cu CARMEN ANA.

```
SQL> UPDATE SALARIATI
  2  SET SALA=SALA*1.15
  3  WHERE FUNCT IN
  4  (SELECT FUNCT
  5  FROM SALARIATI
  6  WHERE NUME='CARMEN ANA');
11 records updated.
```


Scanned with OKEN Scanner

CAPITOLUL 6
DEFINIREA UNEI BAZE DE DATE

Crearea unei baze de date implică, în primul rînd, crearea unei partiţii şi a modelului de alocare a spaţiului în care vor fi memorate datele.

În al doilea rînd, crearea presupune definirea tabelelor, indecşilor, cluster-elor, viziunilor şi sinonimelor utilizate în selectarea şi actualizarea datelor.

Toate aceste operaţii sînt realizate cu ajutorul unui set de comenzi denumite în limbajul SQL ***comenzi pentru definirea datelor.***

6.1. Crearea şi modificarea unei partiţii

O bază de date Oracle este formată din partiţia **SYSTEM** şi, eventual, din alte partiţii, care pot fi create numai de către utilizatorii cu drept de administrare a bazei de date. Comanda folosită pentru crearea de noi partiţii este **CREATE PARTITION.**

Comanda are următoarea sintaxă:

```
CREATE PARTITION nume-de-partiţie;
```

unde:

nume-de-partiţie este numele partiţiei ce urmează a fi creată.

Exemplu:

```
SQL> CREATE PARTITION COMBAZA;
Partition created.
```

Unei partiţii îi sînt asociate mai multe fişiere în care sînt stocate date. Crearea acestor fişiere se face cu ajutorul utilitarului **CCF** (Create Contigues File).

Sintaxa comenzii de apel a utilitarului CCF este:

```
>CCF nume-fişier[.DBS] n [K] [CONTIG]
```


Scanned with OKEN Scanner

unde:

nume-fişier este numele fişierului ce urmează a fi creat;

n este numărul de kiloocteţi alocaţi.

Modificarea unei partiţii trebuie înţeleasă în sensul adăugării de noi fişiere la acea partiţie.

Comanda pentru modificarea partiţiilor este **ALTER PARTITION**, comandă care poate fi folosită numai de către utilizatorii care au fost autorizaţi de către administratorul bazei de date.

Sintaxa acestei comenzi este:

```
ALTER PARTITION nume-partiţie
      ADD FILE nume-fişier;
```

unde:

nume-partiţie este numele partiţiei care se modifică;

nume-fişier este numele fişierului care va fi adăugat partiţiei create.

Exemplu:

```
C:\>ORACLE5\BIN>CCF DEPOZ.DBS 6 K CONTIG
CCF: Version 5.1.17.3 - Production.
CCF: file created
SQL> ALTER PARTITION COMBAZA ADD
  2  FILE 'DEPOZ.DBS';
Partition altered.
```

Prin adăugare de fişiere la o partiţie existentă se sporeşte capacitatea bazei de date.

6.2. Definirea, modificarea şi ştergerea unui model de alocare a spaţiului

La crearea de tabele şi clustere se pot construi modele de alocare a spaţiului, prin comanda **CREATE SPACE**.

Sintaxa ei este:

Scanned with OKEN Scanner

```
CREATE SPACE [DEFINITION] nume-spaţiu
 [DATAPAGES(
   [INITIAL {5 | n}] [INCREMENT {25 | n}]
   [MAXEXTENTS {9999 | n} [PCTFREE {20 | n}]
 )]
 [INDEXPAGES(
   [INITIAL {5 | n}] [INCREMENT {25 | n}]
   [MAXEXTENTS {9999 | n}]
 )]
 [PARTITION {SYSTEM | nume-partiţie}];
```

unde:

nume-spaţiu este numele definiţiei de spaţiu.

DATAPAGES este clauza care controlează alocarea spaţiului pentru o tabelă.

INITIAL specifică numărul de pagini ce se alocă la crearea unei tabele sau a unui cluster. Valoarea minimă a parametrului INITIAL este de 3 pagini iar cea implicită de 5 pagini.

INCREMENT specifică numărul de pagini suplimentare care se vor adăuga în condiţiile în care spaţiul alocat iniţial devine insuficient (cînd tabela/indexul se umple). Valoarea minimă este de 3 pagini iar cea implicită de 25 de pagini.

MAXEXTENTS specifică numărul maxim de alocări suplimentare de spaţiu. Valoarea implicită 9999 are semnificaţia de 'cea mai mare valoare permisă de sistemul de operare gazdă'.

PCTFREE se referă la spaţiul (în procente) care va fi păstrat liber pentru actualizări. Implicit acest procentaj este de 20%.

INDEXPAGES este clauza care controlează alocarea spaţiului pentru indecşi.

INITIAL, INCREMENT şi MAXEXTENTS au aceeaşi semnificaţie ca cea prezentată anterior, cu diferenţa că este vorba de alocare de spaţiu pentru indecşi şi nu pentru tabele.

PARTITION identifică partiţia din baza de date în care va fi alocat spaţiul respectiv.

Cuvîntul cheie **DEFINITION** nu are efect asupra modului de

Scanned with OKEN Scanner

execuţie a comenzii.

Exemple:

1) Să se creeze un model de alocare a spaţiului de dimensiuni reduse, pentru o tabelă/cluster :

```
SQL> CREATE SPACE MINIM
  2  DATAPAGES  (INITIAL 3, INCREMENT 3)
  3  INDEXPAGES (INITIAL 3);
```

Space definition created.

2) Să se creeze un model de alocare a spaţiului de dimensiuni mari pentru o tabelă/cluster:

```
SQL> CREATE SPACE MAXIM
  2  DATAPAGES (INITIAL 100,INCREMENT 20);
```

Space definition created.

Modificarea unui model de alocare a spaţiului se realizează prin intermediul comenzii **ALTER SPACE.**

Sintaxa acestei comenzi este:

```
ALTER SPACE [DEFINITION] nume-spaţiu
 [DATAPAGES(
   [INITIAL {5 | n}] [INCREMENT {25 | n}]
   [MAXEXTENTS {9999 | n}] [PCTFREE {20 | n}]
 )]
 [INDEXPAGES(
   [INITIAL {5 | n}] [INCREMENT {25 | n}]
   [MAXEXTENTS{9999 | n}]
 )]
 [PARTITION {SYSTEM | nume-partiţie}];
```

Semnificaţia parametrilor este identică cu cea prezentată în cazul comenzii de creare a modelului de alocare a spaţiului (**CREATE SPACES**).

Comanda **ALTER SPACES** are ca efect modificarea parametrilor specificaţi în ea (ceilalţi parametri rămîn nemodificaţi). Modificările

Scanned with OKEN Scanner

astfel realizate asupra bazei de date sînt permanente.

Exemplu:

Să se modifice parametrii ce caracterizează spaţiul cu numele MINIM, alocat astfel: numărul de pagini suplimentare 4, iar pentru indecşi 3.

```
SQL> ALTER SPACE MINIM
  2 DATAPAGES  (INCREMENT 4)
  3 INDEXPAGES (INCREMENT 3);
Space altered.
```

Comanda de ştergere a unui model de alocare a spaţiului este **DROP SPACE**. Are următoarea sintaxă:

```
DROP SPACE [DEFINITION] nume-spaţiu;
```

Ca şi la comenzile de creare şi modificare ale unui model de alocare a spaţiului, cuvîntul **DEFINITION** are doar rol de explicitare a comenzii fără a avea vreun efect în execuţia ei.

Cînd se şterge definirea unui model de alocare a spaţiului din baza de date, tabelele şi clusterele create pe baza acestui model nu sînt afectate.

Exemplu:

Să se şteargă spaţiul de alocare cu numele MINIM.

```
SQL> DROP SPACE DEFINITION MINIM;
Space dropped.
```

6.3. Crearea, modificarea şi ştergerea unui cluster

Pentru a avea un acces mai rapid la datele conţinute în baza de date se pot crea *clustere*. Un cluster poate fi definit ca o regrupare fizică de una sau mai multe tabele. Comanda care realizează acest

Scanned with OKEN Scanner

lucru este **CREATE CLUSTER**. Ea are următoarea sintaxă:

```
CREATE CLUSTER nume-cluster
 (spec-col [NOT NULL],spec-col [NOT NULL],...)
 [SIZE n]
 [SPACE nume-spaţiu]
 [COMPRESS | NOCOMPRESS];
```

unde:

nume-cluster este numele pe care utilizatorul îl stabileşte pentru clusterul creat;

spec-col cuprinde numele coloanei, formatul şi mărimea ei. **nume-coloană** este numele coloanei din cluster. Coloana din cluster poate diferi de numele de coloană din tabelă. Pentru o înţelegere mai uşoară, atît coloana din cluster cît şi cea din tabelă pot primi acelaşi nume. **format** reprezintă formatul coloanei din cluster. Mărimea totală a coloanelor din cluster nu poate depăşi 232 de caractere. **mărime** specifică, în octeţi, lungimea blocului logic. Ea trebuie să fie un divizor al lungimii blocului fizic. Dacă valoarea furnizată ca mărime a blocului logic nu este un astfel de divizor, sistemul va folosi o valoare mai mare, care satisface această cerinţă. Dacă dimensiunea blocului logic este mai mare decît cea a blocului fizic, sistemul va utiliza dimensiunea celui din urmă. O valoare prea mică a blocului logic va mări însă timpul necesar de acces la tabela din cluster.

NOT NULL este un parametru opţional. Trebuie să existe cel puţin o coloană careia să i se ataşeze o astfel de opţiune.

SIZE specifică mărimea blocului logic(n), în octeţi.

SPACE defineşte numele de alocare a spaţiului pe disc, pentru cluster.

COMPRESS permite comprimarea indexului coloanei introduse în cluster. Reducerea spaţiului ocupat de index conduce la mărirea vitezei de acces la tabele. **NOCOMPRESS** inhibă comprimarea indexului.

Exemplu:

Să se realizeze un cluster cu numele **BAZA**.

Scanned with OKEN Scanner

```
SQL> CREATE CLUSTER BAZA
  2  (CODP NUMBER(6),
  3  CODC NUMBER(6),
  4  CODD NUMBER(6),
  5  INFORMATII CHAR(21));
Cluster created.
```

Adăugarea sau ştergerea unei tabele din cluster impun modificarea acestuia. Comanda care realizează modificarea unui cluster este **ALTER CLUSTER** care are două sintaxe:

```
ALTER CLUSTER nume-cluster
 ADD TABLE nume-tabelă
 WHERE nume-cluster.nume-col=
       nume-tabelă.nume-col;
ALTER CLUSTER nume-cluster
     DROP TABLE nume-tabelă;
```

Prima sintaxă se utilizează pentru adăugarea unei noi tabele la clusterul existent.

Clauza **WHERE** reprezintă condiţia de clusterizare, ea specificînd coloana din cluster şi coloana din tabelă care se adaugă şi care conţine datele comune care stau la baza realizarii clusterului.

A doua sintaxă se utilizează pentru ştergerea unei tabele dintr-un cluster existent.

Exemple:

1) Să se modifice clusterul definit anterior prin adăugarea tabelei PRODUSE.

```
SQL> ALTER CLUSTER BAZA
  2  ADD TABLE PRODUSE
```


Scanned with OKEN Scanner

```
  3 WHERE BAZA.CODP=PRODUSE.CODP;
Cluster altered.
```

2) Să se şteargă tabela PRODUSE din cluster-ul BAZA.

```
SQL> ALTER CLUSTER BAZA
  2 DROP TABLE PRODUSE;
Cluster altered.
```

Pentru a şterge un cluster din baza de date se foloseşte comanda **DROP CLUSTER**. Ea are următoarea sintaxă:

```
DROP CLUSTER nume-cluster;
```

unde:

nume-cluster reprezintă numele cluster-ului ce trebuie şters.

În urma ştergerii unui cluster, tabelele pe care acesta le conţine nu mai sînt grupate. Tabelele nu sînt afectate prin ştergerea cluster-ului care le conţine.

Cu ajutorul acestei comenzi se pot şterge şi clustere create de alţi utilizatori.

Scoaterea unei tabele individuale dintr-o grupare de tabele (cluster) este mai complicată. Pentru a realiza acest lucru, trebuie creată o nouă tabelă cu aceeaşi structură şi cu acelaşi conţinut ca şi cea veche. Vechea tabelă va fi ştearsă cu **DROP TABLE** şi se va utiliza comanda **RENAME** pentru a da noii tabele numele celei vechi. Dacă vechea tabelă are indecşi, trebuie regăsite definiţiile lor din tabelele dicţionarului de date **INDEXES** sau **SYSINDEXES**. Aceste definiţii se şterg înainte de a şterge vechea tabelă.

Exemplu:

Să se şteargă clusterul BAZA.

```
SQL> DROP CLUSTER BAZA;
Cluster dropped.
```


Scanned with OKEN Scanner

6.4. Crearea, modificarea şi ştergerea unei tabele

Comanda de creare a unei tabele, **CREATE TABLE**, realizează fie definirea unei tabele, urmînd ca introducerea de date să se efectueze ulterior, fie definirea şi încărcarea cu date a unei tabele, chiar în momentul creării ei, prin specificarea unei cereri. Comanda **CREATE TABLE** poate avea două sintaxe:

```
CREATE TABLE nume-tabelă
  (spec-col [NOT NULL],...)
  [SPACE definire-spaţiu [PCTFREE n] |
  CLUSTER nume-cluster (nume-col,...)];
CREATE TABLE nume-tabelă
  [(nume-col [NOT NULL],...)]
  [SPACE definire-spaţiu [PCTFREE n] |
  CLUSTER nume-cluster (nume-col,...)]
  [AS cerere];
```

unde:

spec-col cuprinde *nume-col, format, mărime.*

nume-col este numele coloanei din tabelă.

format reprezintă formatul coloanei din tabelă.Formatele acceptate sînt specificate în tabelul 6.1.

Tabelul 6.1.

Tipul datelor	*Formatul*	*Explicaţii*
Char	CHAR(mărime)	Date alfanumerice.Mărimea maximă a coloanei este de 240 caractere.

Scanned with OKEN Scanner

Tipul datelor	*Formatul*	*Explicații*
Date	DATE	Permit introducerea cîmpurilor de tip dată. Exemplu: January 1, 4712 BC to December 31, 4712 AD
Long	LONG	Date caracter de mărime variabilă. Într-o tabelă trebuie să fie definită doar o coloană de tip LONG.
	LONG VARCHAR	La fel ca LONG.
	LONG ROW	Date binare RAW.
Numerice	NUMBER	Date numerice de mărime implicită.
	NUMBER (mărime)	Date numerice de mărime specificată.
	NUMBER (întregi zecimali)	Date numerice în baza zece.
	NUMBER(*)	La fel ca datele de tip NUMBER.
	DECIMAL	La fel ca datele de tip NUMBER.
	FLOAT	Nu acceptă descrierea mărimii coloanei.
	INTEGER	La fel ca datele de tip NUMBER.
	INTEGER (mărime)	Nu acceptă descrierea mărimii zecimalelor.

Scanned with OKEN Scanner

Tipul datelor	*Formatul*	*Explicaţii*
Numerice	SMALLINT	La fel ca date de tip INTEGER
RAW	RAW(mărimea)	Date binare RAW. Mărimea maximă este 240 octeţi.
	LONG RAW	Date de tip LONG.

În cazul celei de-a doua sintaxe, tipul şi mărimea coloanelor vor fi copiate din rezultatul obţinut în urma specificării **AS cerere**.

Dacă toate coloanele din **cerere** au fost bine definite şi au nume unice, atunci numele coloanelor din noua tabelă creată pot fi omise, căci se vor folosi numele coloanelor din **cerere**.

Dacă se specifică numele coloanelor noii tabele, numărul acestora va trebui să fie egal cu numărul coloanelor din cerere. După ce tabela a fost creată, rezultatele **cererii** sînt introduse în tabelă şi reprezintă rîndurile acesteia.

În ambele sintaxe se utilizează alte cuvinte cheie sau parametrii, care au următoarea semnificaţie:

NULL sau **NOT NULL** se referă la cîmpurile din coloane care pot avea sau nu valori nule. Clauzele **nume-col [NOT NULL]** se pot repeta pentru maximum 254 de coloane.

SPACES desemnează modelul de alocare a spaţiului pentru tabela ce se crează. Modelul este creat în prealabil cu **CREATE SPACES**. Dacă parametrul **SPACE** este omis, pentru toţi parametrii de spaţiu vor fi utilizate valorile implicite din comanda **CREATE SPACE**.

PCTFREE permite specificarea unei alte valori decît cea implicită de 20% pentru spaţiul lăsat liber. Are aceeaşi funcţie ca şi în comanda **CREATE SPACE**.

CLUSTER este parametrul care indică faptul că tabela va fi inclusă în clusterul cu numele **nume-cluster**. Introducerea datelor în tabelă va declanşa memorarea lor în ordinea indicelui de cluster.

Nume-col se referă la numele coloanelor din tabelă care corespund coloanelor din cluster.

Scanned with OKEN Scanner

Exemplu:

Să se creeze tabelele: *PRODUSE, CLIENTI, PRETURI,SALARIATI, DEPOZITE, COMENZI* ce intră în componența bazei de date COMBAZA.

```
SQL> CREATE TABLE PRODUSE
  2   (CODD NUMBER(6) NOT NULL,
  3   CODP NUMBER(5) NOT NULL,
  4   DENP CHAR(11),
  5   STOC NUMBER(6),
  6   DATACRT DATE,
  7   UM CHAR(3));
Table created.
SQL> CREATE TABLE CLIENTI
  2  (CODC NUMBER(6) NOT NULL,
  3  DENC CHAR(11),
  4  LOC CHAR(11),
  5  STR CHAR(16),
  6  NR CHAR(3),
  7  CONT NUMBER(11),
  8  TEL NUMBER(8),
  9  TFAX NUMBER(8));
Table created.
SQL> CREATE TABLE PRETURI
  2  (CODP NUMBER(5) NOT NULL,
  3  PRETMAX NUMBER(9,2),
  4  PRETMIN NUMBER(9,2),
  5  DATAI DATE,
  6  DATASF DATE);
Table created.
SQL> CREATE TABLE SALARIATI
  2  (MARCA NUMBER(4) NOT NULL,
  3  NUME CHAR(16),
  4  FUNCT CHAR(7),
  5  CODD NUMBER(6) NOT NULL,
  6  SALA NUMBER(9,2),
```


Scanned with OKEN Scanner

Apevaloaie D. Daniel

```
  7  VENS NUMBER(9,2),
  8  CODS NUMBER(4) NOT NULL);
Table created.
SQL> CREATE TABLE DEPOZITE
  2  (CODD NUMBER(6) NOT NULL,
  3  DEND NUMBER(6) NOT NULL,
  4  NRSAL NUMBER(2));
Table created.
SQL> CREATE TABLE COMENZI
  2  (NRCOM NUMBER(6) NOT NULL,
  3  CODP NUMBER(5) NOT NULL,
  4  CODC NUMBER(6) NOT NULL,
  5  DATAL DATE,
  6  CANT NUMBER(7),
  7  PRET NUMBER(9.2));
Table created.
```

Modificarea unei tabele presupune adăugarea de noi coloane la sfîrşitul acesteia sau modificarea tipurilor unor coloane deja existente. Comanda care realizează aceste operaţii este **ALTER TABLE**. Sintaxa ei este:

```
ALTER TABLE table {ADD | MODIFY}
      (spec-col [NULL | NOT NULL,...])
```

unde:

ADD adaugă la sfîrşitul tabelei noi coloane care iniţial au valori nule; ulterior, prin actualizare, aceste coloane se vor completa cu date.

MODIFY schimbă definirea unei coloane existente. Modificarea tipului sau mărimii unei coloane presupune ca aceasta să conţină numai valori nule.

spec-col reprezintă numele, formatul şi mărimea coloanei ce se va adăuga sau se va modifica.

NULL se referă la faptul că valorile din coloană ce se adaugă/modifică sînt nule.

NOT NULL specifică faptul că o coloană nu poate avea valori nule (NOT NULL). Unei coloane deja existente i se poate ataşa parametrul

Scanned with OKEN Scanner

NOT NULL numai dacă aceasta conţine valori nenule.

Observaţie: Dacă se adaugă noi coloane la sfirşitul unei tabele referite printr-o viziune, care la rîndul său este definită printr-o cerere care selectează toate coloanele (**SELECT * FROM...**), atunci viziunea nu va mai lucra corect, ea trebuind să fie ştearsă şi recreată.

Exemple:

1) Să se adauge la tabela DEPOZITE o nouă coloană(CANTDEP) ce nu a fost prevăzută iniţial.

```
SQL> ALTER TABLE DEPOZITE ADD
  2  (CANTDEP NUMBER(7));
Table altered.
```

2) Să se modifice atributele coloanei CANTDEP (respectiv mărirea acesteia cu două unităţi).

```
SQL> ALTER TABLE DEPOZITE MODIFY
  2  (CANTDEP NUMBER(9));
Table altered.
```

Pentru ştergerea unei tabele într-o bază de date se utilizează comanda **DROP TABLE**.

Sintaxa ei este:

```
DROP TABLE nume-tabelă;
```

În general, nu se pot şterge tabelele create de alţi utilizatori. La ştergerea unei tabele, se şterg automat indexurile corespunzătoare (create fie de către proprietarul tabelei, fie de către alţi utilizatori) şi privilegiile conferite în legătură cu ea. Rămîn, însă, viziunile şi sinonimele referitoare la tabela ştearsă, care vor deveni invalide. Va trebui fie ca acestea să fie şterse, fie să se definească sau să se redefinească tabela în aşa fel încît viziunile şi sinonimele să devină valide.

Exemplu: Să se şteargă tabela DEPOZITE.

```
SQL> DROP TABLE DEPOZITE;
Table dropped.
```


Scanned with OKEN Scanner

Se pot modifica numele atribuite tabelelor, vizunilor sau sinonimelor utilizînd comanda RENAME. Sintaxa ei este:

```
RENAME nume-vechi TO nume-nou;
```

unde:

nume-vechi şi **nume-nou** sînt constante de tip şir de caractere (nu se scriu între ghilimele sau apostrofuri).

6.5. *Crearea şi ştergerea sinonimelor pentru tabele sau viziuni*

Pentru a simplifica accesul la tabele sau la viziuni se pot crea sinonime. Ele sînt asignate unor specificatori de tabelă foarte lungi, cuprinzînd atît numele tabelei sau al viziunii cît şi numele utilizatorului care le-a creat.

Comanda de creare de sinonime este **CREATE SYNONYM**, cu următoarea sintaxă:

```
CREATE [PUBLIC] SYNONIM nume
       FOR [nume-util]
       nume-tabelă [@bază-de-date];
```

unde:

PUBLIC permite crearea unui sinonim public la care orice utilizator se poate referi fără a mai fi necesară calificarea. Dacă **PUBLIC** este omis, toţi utilizatorii, mai puţin cel care a creat sinonimul, trebuie să califice sinonimul cu numele creatorului. Numai utilizatorii cu drept de administrare a bazei de date pot crea sinonime publice.

nume-util este numele utilizatorului care crează sinonimul. Dacă este omis numele creatorului, **CREATE SYNONYM** presupune că tabela/viziunea aparţine utilizatorului care dă comanda de creare a sinonimului.

nume-tabelă este numele unei tabele, numele unei viziuni sau chiar numele unui alt sinonim.

Scanned with OKEN Scanner

@bază-de-date specifică faptul că sinonimul se referă la o tabelă dintr-o bază de date aflată într-un nod al rețelei.

Ştergerea unui sinonim din baza de date se realizează prin comanda **DROP SYNONYM**:

```
DROP [PUBLIC] SYNONYM nume-sinonim;
```

Exemple:

1) Să se creeze sinonimul DEP pentru tabela DEPOZITE din baza de date COMBAZA1 (prin utilizatorul UTIL).

```
SQL> CREATE SYNONYM DEP
  2       FOR UTIL.DEPOZITE @COMBAZA1;
Synonym created.
```

2) Să se şteargă sinonimul DEPOZITE.

```
SQL> DROP SYNONYM DEPOZITE;
Synonym dropped.
```

6.6. Crearea şi ştergerea unei viziuni

Crearea unei viziuni se realizează cu comanda **CREATE VIEW**. O viziune poate fi creată pornind de la una sau mai multe tabele. Pentru definirea unei viziuni se foloseşte o cerere SQL. Rezultatul oricărei cereri poate fi considerat o tabelă.

Sintaxa comenzii **CREATE VIEW** este:

```
CREATE VIEW nume-viziune
[(nume-col,nume-col...)] AS cerere
[WITH CHECK OPTION];
```

unde:

AS cerere identifică coloanele şi rîndurile tabelelor şi/sau viziunilor care apar în viziunea ce se crează.

Scanned with OKEN Scanner

WITH CHECK OPTION indică faptul că introducerile şi actualizările realizate asupra viziunii nu vor putea crea rînduri pe care viziunea să nu le poată selecta. Această opţiune nu impune restricţii referitoare la cererea folosită în clauza **AS**. De asemenea, **WITH CHECK OPTION**, poate fi folosită într-o viziune ce se bazează pe o altă viziune dar, în acest context, opţiunea va fi ignorată.

nume-col reprezintă numele coloanelor care vor face parte din viziune. Dacă aceste nume nu se specifică, în mod implicit din viziune vor face parte coloanele din cerere. În acest caz, numele coloanelor din cerere trebuie să fie unice.

O viziune defineşte un subset de date memorate în tabelele pe care a fost definită. În continuare se pot selecta subseturi prin introducerea clauzei **WHERE** în cerere.

La crearea unei viziuni nu se poate specifica o clauză **ORDER BY**. Se pot realiza modificări (**UPDATE**) sau ştergeri (**DELETE**) de rînduri din viziune numai dacă viziunea este bazată pe o singură tabelă şi dacă în cerere nu sînt specificate clauzele **GROUP BY**, **DISTINCT**, funcţii de grup sau referiri la pseudocoloană **ROWNUM**.

Se pot introduce rînduri în viziune numai dacă aceasta îndeplineşte restricţiile de mai sus şi cererea nu conţine coloane definite prin expresii aritmetice sau funcţii (deoarece aceste coloane nu sînt memorate fizic în tabelă sînt numite coloane virtuale).

Observaţie: Dacă se adaugă noi rînduri unei tabele referită printr-o viziune care la rîndul ei este definită printr-o cerere care selectează toate coloanele, atunci viziunea nu mai este validă, ea trebuind să fie ştearsă şi recreată.

Ştergerea viziunilor se realizează cu ajutorul comenzii **DROP VIEW**. Sintaxa comenzii este:

```
DROP VIEW nume-viziune;
```

Exemple:

1) Să se creeze viziunea cu numele DEP pe baza coloanelor tabelei DEPOZITE pentru care codul este egal cu 1000.

Scanned with OKEN Scanner

```
SQL> CREATE VIEW DEP AS SELECT *
  2  FROM DEPOZITE
  3  WHERE CODD = 1000;
View created.
```

2) Să se şteargă viziunea DEP.

```
SQL> DROP VIEW DEP
View dropped.
```

6.7. *Crearea, validarea şi ştergerea indecşilor pentru o tabelă*

În vederea obţinerii de performanţe superioare privind accesul la tabelele unei baze de date se pot construi indecşi folosind comanda **CREATE INDEX.**

Comanda are ca efect crearea unui index la o tabelă, index care va conţine cîte o intrare pentru fiecare valoare care apare în coloana specificată pentru tabela respectivă. Această coloană se numeşte coloana de index.

Sintaxa comenzii CREATE INDEX este:

```
CREATE [UNIQUE] INDEX nume-index
   ON nume-tabelă (nume-col [ASC|DESC],
                   nume-col [ASC|DESC],...)
   [COMPRESS | NOCOMPRESS]
   [SYSSORT  | NOSYSSORT]
   [ROWS = n] [PCTFREE = {20 | n}];
```

unde:

UNIQUE se referă la faptul că tabela nu va conţine două rînduri cu aceleaşi valori în coloanele index (deci nu vor exista dubluri în coloana index). Dacă se omite, tabela poate conţine oricîte asemenea rînduri care să aibă în coloana index aceeaşi valoare.

ASC|DESC precizează ordinea în care va fi păstrat indexul:

Scanned with OKEN Scanner

ascendentă sau descendentă.

COMPRESS indică faptul că indecşii pot fi comprimaţi, reducîndu-se în acest mod spaţiul necesar memorării acestora şi mărindu-se viteza operaţiilor care folosesc indecşii. Comprimarea nu permite găsirea într-un index a valorii unui anumit cîmp fără să acceseze tabela. **NOCOMPRESS** suprimă comprimarea indecşilor.

SYSSORT specifică faptul că procedura standard ORACLE de sortare este folosită pentru a crea un index. **NOSYSSORT** este folosit pentru depanare.

ROWS specifică numărul aproximativ de rînduri ce urmează a fi indexate. Acest număr este utilizat pentru a optimiza sortarea în SYSSORT. Clauza nu are efect în SQL*PLUS.

PCTFREE specifică, în momentul creării indexu-lui, procentul din spaţiul pentru index ce trebuie să rămînă liber pentru actualizări. Nu are efect asupra extensiilor alocate după crearea indexului.

Indexul creat permite creşterea vitezei de acces la datele unei tabele prin intermediul coloanei index datorită faptului că se vor căuta rîndurile corespunzătoare valorilor coloanei index, fără a citi întreaga tabelă. Deoarece indexul şi tabela trebuie actualizate, introducerile, ştergerile şi modificările în cîmpul de valori ale coloanei index necesită un timp mai mare. De aceea, se recomandă crearea de maximum 3 indexuri pentru o tabelă. În funcţie de necesităţi, se pot crea şi şterge în mod dinamic indecşi şi se poate experimenta dacă este sau nu cazul să se creeze indecşi pentru obţinerea de performanţe cît mai bune.

Observaţie: Indecşii creaţi pentru o singură tabelă trebuie să aibă nume distincte. Pentru tabele diferite pot fi creaţi indecşi cu acelaşi nume. În momentul ştergerii acestor indecşi trebuie specificată tabela din care fac parte.

Comanda de validare a unui index este **VALIDATE INDEX**. Această comandă verifică integritatea indexului specificat pentru o anumită tabelă. Dacă pe tabela respectivă a fost creat un singur index, al cărui nume va fi specificat în comandă, atunci numele tabelei poate fi omis. Comanda **VALIDATE INDEX** are sintaxa:

```
VALIDATE INDEX nume-index
        [ON nume-tabelă] [WITH LIST];
```


Scanned with OKEN Scanner

unde:

WITH LIST salvează informaţiile referitoare la index într-un fişier.

nume-index este numele indexului ce urmează a fi validat. Dacă indexul este valid, comanda va afişa un mesaj corespunzător: **Index validated.** Obţinerea oricărui alt mesaj va determina ştergerea indexului şi recrearea lui.

Ştergerea unui index din baza de date se realizează cu ajutorul comenzii **DROP INDEX**, cu sintaxa:

```
DROP INDEX nume-index [ON nume-tabelă];
```

Dacă pe o anumită tabelă a fost creat un singur index, numele tabelei respective poate fi omis în linia de comandă. Este posibil ca doi indecşi creaţi pe tabele diferite să aibe acelaşi nume. În acest caz, în linia de comandă vor apărea atît numele indexului cît şi al tabelei pentru care a fost creat. Se recomandă numai ştergerea indecşilor proprii.

Exemple:

1) Să se creeze un index cu numele SUMAR pentru coloana FUNCT din tabela SALARIATI.

```
SQL> CREATE INDEX SUMAR
  2  ON SALARIATI(FUNCT);
Index created.
```

2) Să se creeze un index cu numele MARCA pentru coloana MARCA din tabela SALARIATI.

```
SQL> CREATE INDEX MARCA
  2  ON SALARIATI(MARCA);
Index created.
```

3) Să se creeze un index cu numele MARCA din pentru coloana MARCA din tabela SALARIATI cu suprimarea comprimării lui.

```
SQL> CREATE INDEX MARCA
  2  ON SALARIATI(MARCA) NOCOMPRESS;
Index created.
```


Scanned with OKEN Scanner

4) Să se verifice integritatea indexului SUMAR.

SQL> VALIDATE INDEX SUMAR;
Index validated.

5) Să se verifice integritatea indexului MARCA pentru tabela SALARIATI.

SQL> VALIDATE INDEX MARCA ON SALARIATI;
Index validated.

6) Să se şteargă indexul SUMAR.

SQL> DROP INDEX SUMAR;
Index dropped.

2) Să se şteargă indexul MARCA din tabela SALARIATI.

SQL> DROP INDEX MARCA ON SALARIATI;
Index dropped.

Scanned with OKEN Scanner

CAPITOLUL 7
CONTROLUL ACCESULUI LA BAZA DE DATE

Accesul la o bază de date se face sub control, utilizînd un **nume de utilizator** şi o **parolă**. De asemenea, sistemul ORACLE, prin intermediul limbajului SQL, permite stabilirea unor privilegii pentru fiecare tabelă din baza de date şi pentru fiecare utilizator. Aceste privilegii pot fi acordate sau retrase doar de cei care au acest drept. Totodată, aceştia pot schimba cuvintele de acces la baza de date.

7.1. Acordarea drepturilor de acces

Acordarea drepturilor de acces la o bază de date se face cu ajutorul comenzii **GRANT**. Ea este utilizată pentru definirea numelor şi parolelor de identificare a unui utilizator, a drepturilor acestuia la baza de date, precum şi pentru acordarea unor drepturi de acces la tabele şi viziuni.

Comanda **GRANT** are următoarele două forme sintactice:

```
GRANT [CONNECT,] [RESOURCE,] [DBA]
   TO nume-util1,nume-util2...
   [IDENTIFIED BY parolă1, parolă2,...];
GRANT [privilegiu, privilegiu,... | ALL]
   ON nume-tabelă
   TO {nume-util1, nume-util2,... | PUBLIC}
   [WITH GRANT OPTION];
```

unde:

CONNECT conferă utilizatorului dreptul de a se conecta la baza de date.

RESOURCE conferă utilizatorului dreptul de a crea tabele şi alte elemente ale bazei de date.

DBA conferă utilizatorului dreptul de administrare a bazei de date.

IDENTIFIED BY precizează parola care va fi furnizată de utilizatori la conectare.

nume-util1, nume-util2 sînt numele utilizatorilor cărora li se acordă drepturi de acces la baza de date.

Scanned with OKEN Scanner

parolă1, parolă2 sînt parolele atribuite utilizatorilor.

privilegiu precizează acţiunile pe care utilizatorul are autoritatea să le execute (vezi tabelul 7.1.)

Tabelul 7.1.

PRIVILEGIU	*FUNCŢII*
ALTER	Modificarea coloanelor tabelei
DELETE	Ştergerea datelor tabelei
INDEX	Indexarea coloanelor tabelei
INSERT	Introducerea datelor în tabele
SELECT	Selectarea de date din tabelă
UPDATE	Actualizarea datelor din tabelă

ALL va conferi toate privilegiile celui care dă comanda. În cazul în care privilegiile se acordă în legătură cu o viziune, acestea pot fi **DELETE, INSERT, SELECT, UPDATE.**

PUBLIC se referă la faptul că privilegiile vor fi conferite tuturor utilizatorilor.

WITH GRANT OPTION oferă posibilitatea ca utilizatorul care a obţinut nişte privilegii să le poată conferi la rîndul său altor utilizatori.

Exemple):

1) Să se definească pentru utilizatorul COMBAZA1 parola de acces UTIL.

```
SQL> GRANT CONNECT TO COMBAZA1
  2  IDENTIFIED BY UTIL;
Grant succeeded.
```

2)· Să se definească un acces privilegiat pentru tabela SALARIATI utilizatorului COMBAZA2.

Scanned with OKEN Scanner

```
SQL> GRANT ALL ON SALARIATI TO COMBAZA2.
Grant succeeded.
```

7.2. Revocarea drepturilor de acces

Drepturile de acces la o bază de date pot fi revocate, utilizînd comanda **REVOKE**. Comanda are două forme sintactice:

```
REVOKE [CONNECT,] [RESOURCE,] [DBA]
    FROM nume-util1,nume-util2,...;
REVOKE {privilegiu,privilegiu,... | ALL}
    ON tabelă
    FROM {nume-util1,nume-util2,... | PUBLIC};
```

Clauzele şi numele de date utilizate în aceste forme sintactice au aceeaşi semnificaţie cu cele din comanda GRANT.

Exemple:

1) Să se revoce drepturile de acces la baza de date COMBAZA1.

```
SQL> REVOKE CONNECT FROM COMBAZA1;
Revoke succeeded.
```

2) Să se revoce accesul privilegiat la tabela PRODUSE utilizatorului UTIL2.

```
SQL> REVOKE ALL ON PRODUSE FROM UTIL2;
Revoke succeeded.
```

7.3. Reţinerea în baza de date a comenzilor utilizator

Comanda **AUDIT** realizează controlul acţiunilor întreprinse asupra elementelor unei baze de date.

Comanda are două forme sintactice: una pentru acţiunile

Scanned with OKEN Scanner

întreprinse asupra tabelelor şi viziunilor şi una pentru utilizarea facilităţilor sistemului:

```
AUDIT {opţiune,opţiune,... | ALL}
      ON {nume-tabelă | DEFAULT}
      [BY {ACCESS | SESSION}]
      [WHENEVER [NOT] SUCCESSFUL];
AUDIT {opţiune,opţiune,... | ALL }
      [WHENEVER [NOT] SUCCESSFUL];
```

Semnificaţia cuvintelor din prima formă sintactică este:

opţiune reprezintă comanda care este luată în considerare referitor la tabela specificată de **nume-tabelă**. Poate fi una din comenzile: **ALTER, AUDIT, CONNENT, DELETE, GRANT, INDEX, INSERT, LOCK, RENAME, UPDATE.**

Opţiunea **GRANT** presupune şi opţiunea **REVOKE.**

ALL presupune luarea în considerare a tuturor comenzilor enunţate anterior.

DEFAULT specifică faptul că acţiunea comenzii **AUDIT** se va răsfrînge asupra tuturor comenzilor nou create. Pentru a putea folosi opţiunea **ON DEFAULT**, utilizatorul trebuie să fie autorizat de către administratorul bazei de date.

BY ACCESS sau **BY SESSION** iau în considerare fiecare comandă sau fiecare sesiune de accesare în legătură cu tabela specificată din **nume-tabelă.**

WHENEVER [NOT] SUCCESSFUL realizează scrierea unui rînd în tabela de verificat, numai dacă accesul la aceasta sau la o facilitate a sistemului a reuşit. Cînd clauza este omisă, rîndul va fi scris, indiferent dacă accesul a reuşit sau nu.

A doua formă sintactică permite verificarea modului de utilizare a facilităţilor sistemului. Folosirea acestei forme, necesită, în prealabil, acordul administratorului bazei de date.

Semnificaţia cuvintelor din a doua formă sintactică este:

opţiune specifică facilităţile care vor fi verificate. Ele pot fi:

CONNECT verifică I/E din baza de date;

DBA verifică comenzile care necesită autorizare din partea DBA;

NOT EXIST verifică toate comenzile care pot genera o eroare de

Scanned with OKEN Scanner

tipul "does not exist";

RESOURCE verifică comenzile care se referă la crearea sau şterge-rea de tabele, clustere, viziuni, indecşi, modele de alocare a spaţiului, sinonime;

ALL verifică toate facilităţile enunţate anterior.

WHENEVER [NOT] SUCCESSFULL are aceeaşi semnificaţie ca la prima sintaxă.

Cînd tabelele sînt apelate printr-un fişier de legătură la baza de date, verificarea este făcută de către sistemul care realizează legătura, după ce se verifică setul de opţiuni ale sistemului respectiv.

Exemple:

1) Să se controleze operaţia de introducere a datelor în tabela SALARIATI.

```
SQL> AUDIT INSERT
  2  ON SALARIATI BY ACCES;
```

Audit succeeded.

2) Să se controleze toate operaţiile realizate asupra datelor din tabela SALARIATI.

```
SQL> AUDIT ALL ON SALARIATI
  2  WHENEVER NOT SUCCESFUL;
```

Audit succeeded.

3) Să se verifice comenzile care se referă la crearea sau ştergerea de tabele, clustere etc.

```
SQL> AUDIT RESOURCE
  2  WHENEVER SUCCESSFUL;
```

Audit succeeded.

7.4. Renunţarea la reţinerea în bază a comenzilor utilizator

Comanda NOAUDIT permite renunţarea parţială sau completă la reţinerea în baza de date a accesului utilizatorilor la anumite tabele sau resurse.

Comanda prezintă două forme sintactice:

Scanned with OKEN Scanner

```
NOAUDIT {opţiune1,opţiune2,... | ALL}
      ON {nume-tabelă | DEFAULT}
      [WHENEVER [NOT] SUCCESSFUL];
NOAUDIT {opţiune1,opţiune2,... | ALL}
      [WHENEVER [NOT] SUCCESSFUL];
```

Toate opţiunile şi clauzele au aceeaşi semnificaţie ca în cazul comenzii **AUDIT**.

Exemple:

1) Să se renunţe la reţinerea în baza de date a accesului utilizatorilor referitor la crearea/ştergerea de tabele, clustere etc.

```
SQL> NOAUDIT RESOURCE
  2  WHENEVER SUCCESSFUL;
Noaudit succeeded.
```

2) Să se renunţe la reţinerea în baza de date a accesului utilizatorilor, referitor la introducerea de date în tabela SALARIATI.

```
SQL> NOAUDIT INSERT ON PRODUSE;
Noaudit succeeded.
```

3) Să se renunţe la reţinerea în baza de date a accesului utilizatorilor, referitor la toate operaţiile realizate asupra datelor din tabela PRODUSE.

```
SQL> NOAUDIT ALL ON PRODUSE
  2  WHENEVER NOT SUCCESFUL;
Noaudit succeeded.
```

7.5. Partajarea accesului la tabele

În mod normal, cînd un utilizator introduce, actualizează sau şterge rînduri dintr-o tabelă, alţi utilizatori nu vor putea modifica tabela pînă cînd nu vor da comanda **COMMIT** sau **ROLLBACK**.

Comanda **LOCK TABLE** dă posibilitatea utilizatorului să împartă

Scanned with OKEN Scanner

în mod egal cu alţi utilizatori accesul la una sau mai multe tabele, păstrînd integritatea acestora.

Sintaxa comenzii este:

```
LOCK TABLE nume-tabelă1, nume-tabelă2,...
    IN {SHARE | SHARE UPDATE | EXCLUSIVE}
    MODE [NOWAIT];
```

unde:

MODE se referă la modul de acces la tabela/tabele de către diferiţi utilizatori.

SHARE este modul care permite celorlalţi utilizatori să interogheze tabela, fără a o modifica.

SHARE UPDATE permite actualizarea acelor rînduri ale tabelei care au fost blocate dar nu au fost şi modificate de alţi utilizatori.

EXCLUSIVE permite actualizarea tuturor rîndurilor unei tabele, interzicînd altor utilizatori să blocheze sau să actualizeze vreun rînd din tabelă.

NOWAIT returnează controlul utilizatorului care a dat comanda de blocare permiţînd introducerea altor comenzi pînă cînd tabela devine disponibilă.

Exemplu:

```
SQL> LOCK TABLE SALARIATI
  2  IN EXCLUSIVE MODE NOWAIT;
Table(s) Locked.
```


Scanned with OKEN Scanner

CAPITOLUL 8
DEFINIREA OPŢIUNILOR DE LUCRU ŞI REALIZAREA ALTOR OPERAŢII

La începutul sau în timpul sesiunii de lucru cu SQL*PLUS se pot schimba opţiunile de supraveghere a execuţiei diferitelor comenzi.

8.1. Setarea opţiunilor de lucru

Pentru a preciza opţiunile de lucru cu SQL*PLUS se utilizează comanda **SET**. Ea are două forme sintactice: una se foloseşte ca *Help* pentru utilizator (se afişează numele variabilelor de sistem în care se memorează diferite opţiuni), iar cealaltă se foloseşte pentru setarea variabilelor de sistem.

O variabilă de sistem este un cîmp rezervat în care se reţine o valoare de către sistemul **ORACLE**. Această valoare indică o stare a sistemului. Valorile variabilelor de sistem pot fi schimbate cu ajutorul comenzii **SET** şi pot fi afişate cu ajutorul comenzii **SHOW**.

Formele sintactice ale comenzii SET sînt:

```
SET {?|HELP};
SET {AUTO[COMIT]
  {OFF|ON|IMM[DIATE]}|ARRAYSIZE {20|n}
  BUF[FER] buffer |
  CMDS[EP]{;|caracter|OFF|ON}|
  CONCAT{.|caracter|OFF|ON}|
  COPYCOMMIT{0|n}
  DCL[SEP] {!|caracter}|
  DEF[INE] {&|caracter |OFF|ON}|
  DOC[UMENT] {OFF|ON}  |
  ECH[OSET]  {OFF|ON}  |
  EMBEDDED {OFF|ON}|
  ESCAPE {caracter|OFF|ON}
```


Scanned with OKEN Scanner

```
FEED[BACK] {6|număr-înreg.|OFF|ON}
FLUSH {OFF|ON}
HEA[DING] {OFF|ON}|
HEADS[EP] {'|'|caracter|OFF|ON}|
LIN[ESIZE] {80|n}|
LONG {80|n} |
MAXD[ATA] n |
NEWP[AGE] {1|n} |
NULL [text] |
NUMF[ORMAT] text |
NUM[WIDTH] {10|n} |
PAGES[IZE] {14|n} |
PAU[SE] {OFF|ON|text} |
SCAN {OFF|ON} |
SHOW [MODE] {OFF|ON} |
SPA[CE] {1|n} |
SQLC[ONTINUE] {'>'|caracter} |
SQLN[UMBER] {OFF|ON} |
SQLPRE[FIX] {#|caracter} |
SQL{[ROMPT] {SQL>|text} |
SQLT[ERMINATOR]{;|caracter|OFF|ON} |
SUFFIX{SQL>|text}
TAB {OFF|ON} |
TERM[OUT] {OFF|ON} |
TI[ME] {OFF|ON} |
TIMI[NG] {OFF|ON} |
TRIM[OUT] {OFF|ON} |
TRU[NCATE] {OFF|ON} |
UND[ERLINE] {-|caracter|OFF|ON} |
VER[IFY] {OFF|ON} |
WRAP {OFF|ON} };
```


Scanned with OKEN Scanner

unde:

AUTO[COMMIT] {OFF | ON | IMM} determină ca, în cazul cînd se utilizează **ON** sau **IMM**, modificările făcute să se răsfrîngă automat asupra bazei de date fizice, iar, în cazul în care se utilizează **OFF**, modificările să fie reţinute separat (neoperîndu-se fizic asupra bazei de date). Pentru ca modificările să se facă fizic asupra bazei de date, trebuie să se dea comanda **COMMIT**, iar dacă se doreşte renunţarea la ele, comanda **ROLLBACK**.

ARRAYSIZE {20 | n} setează numărul de rînduri din baza de date care pot fi prelucrate la un moment dat.

BUF[FER] buffer permite schimbarea buffer-ului curent cu buffer-ul specificat în comandă. Iniţial buffer-ul SQL este buffer-ul curent.

CMDS[EP] {; | caracter | OFF | ON} defineşte caracterul care serveşte drept separator între diferitele comenzi aflate pe aceeaşi linie. Implicit acesta este '|', dar poate fi orice alt caracter ASCII. Opţiunile **ON** şi **OFF** precizează dacă se acceptă sau nu introducerea mai multor comenzi pe aceeaşi linie.

CONCAT {. | caracter | OFF | ON} setează un caracter (altul decît '.' care este implicit) care poate fi folosit la calificări, în referirile la variabilele utilizator şi care, în mod normal, ar putea fi interpretat ca o parte din nume. Prin opţiunea **ON** se face o resetare (se revine la funcţia implicită a caracterului '.') iar prin opţiunea **OFF** se declară că nu se vor folosi caractere de concatenare.

COPYCOMMIT {0 | n} permite lansarea unei comenzi **COMMIT** în timpul copierii numărului, n, de rînduri de la o bază de date la alta. Valorile valide pentru numărul de rînduri sînt cuprinse între 0 şi 5000. Valoarea 0 va determina lansarea comenzii COMMIT la sfîrşitul operaţiei de copiere.

DCL[SEP] {! | caracter} defineşte caracterul care va fi utilizat pentru separarea mai multor comenzi ale sistemului de operare, introduse în timpul lucrului cu SQL*PLUS. ORACLE recunoaşte comenzile sistemului de operare prin faptul ca acestea încep cu **$**.

DEF[INE] {& | caracter | OFF | ON} defineşte caracterul care va fi utilizat pentru prefixarea variabilelor de substituţie. Valorile **OFF** şi **ON** determină înlocuirea sau nu a variabilelor de substituţie din comenzi cu valorile date de utilizator.

DOC[UMENT] {OFF | ON} defineşte validitatea sau invaliditatea comenzii DOCUMENT. **ON** consideră comanda validă, în timp ce **OFF**

Scanned with OKEN Scanner

consideră comanda invalidă şi drept urmare textul care urmează după ea va fi tratat ca o comandă. Sfîrşitul textului document se marchează prin caracterul # pe o linie separată.

ECH[OSET] {OFF|ON} precizează dacă textul corespunzător unui fişier de comenzi va fi afişat pe ecran sau nu. **ON** va permite afişarea comenzilor iar **OFF** va inhiba afişarea acestora.

EMBEDDED {OFF|ON} stabileşte modul în care începe afişarea unui raport, în urma execuţiei unei comenzi neprocedurale. **OFF** forţează începerea raportului pe o pagina nouă iar **ON** determină începerea raportului oriunde pe pagină.

ESCAPE {caracter|OFF|ON} defineşte un caracter "**escape**". Acesta, pus în faţa caracterului care desemnează o variabilă de substituţie, va determina considerarea caracterului respectiv drept unul obişnuit (fără funcţie de escape). **OFF** suprimă definirea caracterului "escape" iar **ON** desemnează caracterul "escape" ca fiind '\'.

FEED[BACK] {6|număr-înreg.|OFF|ON} permite afişarea numărului de înregistrări selectate în urma unei cereri, în cazul în care acesta este mai mare sau egal cu **număr-înreg** (repectiv **6**). Dacă se utilizează **ON** se setează numărul de înregistrări la 1 (se va afişa întotdeauna numărul de înregistrări selectate) iar dacă se utilizează **OFF** nu se va mai afişa numărul înregistră-rilor selectate.

FLUSH {OFF|ON} permite vizualizarea (**ON**) sau inhibarea (**OFF**) vizualizării textelor care rezultă în urma execuţiei comenzilor.

HEA[DING] {OFF|ON} permite afişarea (**ON**) sau nu (**OFF**) a capetelor de tabel (numelor de coloane) în rapoarte.

HEADS[EP] {'|'|caracter|OFF|ON} desemnează caracterul care va fi utilizat ca separator pentru titlurile rapoartelor. La întîlnirea acestui caracter, textul care urmează va fi afişat sau tipărit pe o nouă linie. Se utilizează în BTITLE şi TTITLE. În mod impli-cit se utilizează caracterul | iar opţiunile **ON** şi **OFF** permit recunoaşterea sau nu a funcţiei de separator a caracterului. În cazul utilizării opţiunii **OFF**, caracterul va fi tratat ca un caracter normal fără a mai avea rolul de separator în titlurile de rapoarte.

LIN[ESIZE] {80|n} precizează numărul de caractere care vor fi afişate pe o linie. Valoarea maximă a numărului de caractere este 999.

LONG {80|n} specifică dimensiunea maximă pentru afişarea şi copierea cîmpurilor de tip LONG. Valorile acceptate aparţin intervalului 1 ÷ 32767.

MAXD[ATA] n specifică dimensiunea totală maximă a unui rînd

Scanned with OKEN Scanner

pe care SQL*PLUS îl poate procesa. Valoarea implicită precum şi cea maximă diferă de la un sistem de operare la altul (pentru detalii, vezi ghidul de instalare).

NEWP[AGE] {1|n} specifică numărul de linii albe care vor fi lăsate între textul scris cu BTITLE şi cel scris cu TTITLE de pe pagina următoare. Valoarea 0 va determina trimiterea caracterului **FORM FEED** între pagini şi va realiza, pentru cele mai multe tipuri de terminale, ştergerea ecranului.

NULL [text] precizează textul care va fi afişat pentru reprezentarea valorilor nule. Dacă nu se specifică textul se va afişa caracterul spaţiu.

NUMF[ORMAT] text specifică, prin **text**, formatul afişării datelor.

NUM[WIDTH] {10|n} specifică dimensiunea implicită pentru afişarea numerelor.

PAGES[IZE] {14|n} determină numărul de linii afişate pe o pagină.

PAU[SE] {OFF|ON|text} specifică dacă la afişare, între pagini se face pauză (aşteptîndu-se tastarea lui **<CR>**) sau nu. **OFF** determină afişarea paginilor fără pauze. **ON** determină aşteptarea confirmării (prin <CR>) pentru trecerea la o pagină nouă. **text** precizează textul care va fi afişat atunci cînd se va face pauză.

SCAN {OFF|ON} precizează dacă să se facă căutarea variabilelor de substituţie şi să li se atribuie noi valori (**ON**) sau să se inhibe acest proces (**OFF**).

SHOW[MODE] {OFF|ON} specifică dacă la schimbarea valorii unei variabile de sistem să se afişeze atît vechea cît şi noua valoare (**ON**), sau să nu se afişeze nimic după schimbare (**OFF**).

SPA[CE] {1|n} specifică numărul de spaţii lăsat între coloane la afişarea unei tabele.

SQLC[ONTINUE] {'>'|caracter} precizează caracterul care va fi afişat ca prompter, în cazul continuării comenzii pe mai multe linii.

SQLN[UMBER] {OFF|ON} specifică, în cazul în care este **ON**, faptul că prompterul care corespunde continuării unei comenzi pe mai multe linii este chiar numărul de linie. **OFF** inhibă afişarea numărului de linie.

SQLPRE[FIX] {#|caracter} specifică caracterul de prefixare a comenzilor SQL. În timpul introducerii unei comenzi specifice SQL*PLUS, la prompterul **SQL>**, se poate introduce şi o altă comandă

SQL, pe o linie separată, prefixată cu caracterul definit prin SQLPREFIX, fără a afecta comanda SQLPLUS definită la promptul **SQL>**.

SQLP[ROMPT] {SQL> | text} setează prompterul care se afişează pentru comenzile SQL, atunci cînd SQLNUMBER este **OFF**. În mod implicit se utilizează textul **SQL**, dar se poate specifica orice alt text.

SQLT[ERMINATOR] {; | caracter | OFF | ON} precizează caracterul utilizat la terminarea comenzilor SQL. Implicit se utilizează caracterul ';', care va fi selectat şi de opţiunea **ON**. **OFF** specifică faptul că nu există un caracter pentru terminarea comenzilor, aceastea fiind recunoscute prin introducerea unor linii goale.

SUFFIX {SQL | text} selectează tipul (extensia) fişierelor de comenzi utilizate în comenzi SQL. În mod implicit se consideră extensia **SQL**.

TAB {OFF | ON} specifică transferarea efectivă a caracterului **TAB** la ieşire (**ON**), sau transformarea acestuia într-un număr corespunzator de spaţii (**OFF**). Valoarea implicită pentru TAB este dependentă de sistem (pentru aflarea ei se poate folosi comanda **SHOW TAB**).

TERM[OUT] {OFF | ON} permite inhibarea afişării pe ecran a rezultatelor generate în urma execuţiei unui fişier de comenzi, ieşirile putînd fi reţinute într-un fişier de tip SPOOL (**ON**). **OFF** permite afişarea pe ecran a ieşirilor.

TI[ME] {OFF | ON} permite afişarea orei curente înaintea fiecărui prompter (**ON**) sau suprimarea afişării acesteia (**OFF**).

TIMI[NG] {OFF | ON} permite evaluarea timpului necesar execuţiei comenzilor SQL (**ON**). **OFF** inhibă acţiunea precizată anterior.

TRIM[OUT] {OFF | ON} afişează sau nu spaţiile care apar la sfîrşitul fiecărei linii. **ON** inhibă afişarea spaţiilor (nesemnificative), ceea ce poate mări viteza operaţiilor de ieşire. **OFF** va afişa spaţiile de la sfîrşitul liniilor. Această comandă nu afectează ieşirile în fişiere.

TRU[NCATE] {OFF | ON} realizează sau nu trunchierea textului care este afişat într-o coloană ale cărei dimensiuni sînt prea mici. **ON** permite trunchierea textului, iar **OFF** va afişa textul care nu încape, în aceeaşi coloană pe următoarele linii. Această comandă este complementară comenzii WRAP, care se preferă datorită faptului că este recunoscută de comanda SHOW, în timp ce TRUNCATE nu este recunoscută.

UND[erline] {- | caracter | OFF | ON} precizează caracterul care va

Scanned with OKEN Scanner

fi folosit pentru sublinierea titlurilor în rapoarte. ON şi OFF permit şi respectiv inhibă sublinierea, fără a afecta caracterul ales pentru subliniere.

VER[IFY] {OFF|ON} afişează sau nu comenzile SQL după ce s-au atribuit variabilelor de substituţie noile valori. ON va afişa textul comenzii înainte şi după înlocuirea referirilor la variabile de substituţie cu noile valori, iar OFF va inhiba afişarea.

WRAP{OFF|ON} va declanşa afişarea textului care nu încape în spaţiul destinat pe liniile următoare în cazul în care se foloseşte ON. Dacă se utilizeaza OFF, va declanşa trunchierea textului care depăşeşte limitele spaţiului destinat.

Dacă se doreşte ca numai pentru anumite coloane să se facă trunchiere iar pentru altele nu, se va utiliza comanda COLUMN împreună cu una din clauzele WRAPPED sau TRUNCATED pentru fiecare coloană în parte.

Exemplu:

Să se listeze setările din sistem.

```
SQL> SET ?
heading ON
linesize 70
numwidth 7
spool ON
user is "SYSTEM"
space 1
worksize DEFAULT
lines will be wrapped
pagesize 25
showmode BOTH
pause is OFF
ttitle OFF and is the 1st few characters of the next SELECT
statement
btitle OFF and is the 1st few characters of the next SELECT
statement
define "&" (hex 26)
escape OFF
```


Scanned with OKEN Scanner

```
concat "." (hex 2e)
trigger OFF
sqlprompt "SQL> "
underline "-" (hex 2d)
null ""
verify ON
message ON
tab ON
scan ON
dclsep OFF
termout ON
echo OFF
sqlcase MIXED
case MIXED
headsep "|" (hex 7c)
maxdata 32767
time OFF
cmdsep OFF
xisql OFF
sqlterminator ";" (hex 3b)
sqlprefix "#" (hex 23)
release 50117
sqlnumber ON
autocommit IMMEDIATE
newpage 1
long 80
document ON
errorflag OFF
trimout ON
timing ON
qbidebug OFF
numformat ""
synonym OFF
suffix "SQL"
flush ON
sqlcontinue "> "
pno 1
```


Scanned with OKEN Scanner

```
lno 9
buffer 54
embedded OFF
arraysize 20
copycommit 0
```

8.2. Afişarea valorilor variabilelor de sistem

Afişarea valorilor variabilelor de sistem se face cu comanda **SHOW**. Ea are două forme sintactice: una este utilizată ca HELP pentru utilizator (se afişează o listă a opţiunilor de bază) iar cealaltă este utilizată pentru afişarea diferitelor valori ale variabilelor sistem. Cele două forme sintactice sînt:

```
SHOW {HELP|?};
SHOW { ALL | BTI[TLE] | LNO | PNO |
       RE[LEASE] | SPOO[L] | SQLCODE |
       SQLERRM | TTI[TLE] | USER | WRAP
       opţiune-setată };
```

unde:

ALL determină afişarea setărilor făcute pentru toate opţiunile;

BTI[TLE] determină afişarea definiţiei pentru titlurile care apar în partea de jos a raportului;

LNO afişează numărul liniei curente, respectiv poziţia în cadrul paginii curente a ieşirilor afişate sau memorate într-un fişier;

PNO afişează numărul paginii curente;

RE[LEASE] afişează numărul de identificare a versiunii;

SPOO[L] arată dacă ieşirile sînt trimise într-un fişier sau nu;

SQLCODE afişează codul de retur al celei mai recente operaţii SQL executate, care se găseşte în variabila de sistem **SQL.SQLCODE**;

SQLERRM afişează textul erorii descrise de SQLCODE, care este memorată în variabila **SQL.SQLERRM**;

Scanned with OKEN Scanner

TTI[TLE] afişează definiţia pentru rîndurile care apar în partea de sus a raportului;

USER arată numele de identificare a utilizatorului;

WRAP arată dacă facilitatea de trunchiere este activată sau nu;

opţiune-setată desemnează orice opţiune care a fost setată prin comanda SET, cu excepţia opţiunii TRUNC.

Exemple:

1) Să se afişeze numărul poziţiei liniei în cadrul paginii curente.

```
SQL> SHOW LNO;
lno 9
```

2) Să se afişeze ultima definiţie a titlurilor ce sînt afişate la sfîrşitul rapoartelor.

```
SQL> SHOW BTI
btitle OFF and is the 1st few characters of the next SELECT
statement
```

3) Să se afişeze dacă ieşirile sînt transmise într-un fişier.

```
SQL> SHOW SPOO
spool ON
```

4) Să se afişeze numele de identificare a utilizato-rului curent.

```
SQL> SHOW USER
user is "SYSTEM"
```

5) Să se afişeze numărul paginii curente.

```
SQL> SHOW PNO
pno 1
```

8.3. Evaluarea timpului necesar executării comenzilor

Pentru a evalua şi afişa timpul, necesar, de cele mai multe ori, pentru analiza performanţelor execuţiei diferitelor comenzi SQL sau fişierului de comenzi, se utilizează comanda TIMING.

Ea are următoarea sintaxă:

Scanned with OKEN Scanner

TIMI[NG] [START text | STOP | SHOW];

unde:

START crează o zonă de contorizare a timpului şi consideră textul ca titlu al acestei zone. Pot exista, în acelaşi timp, mai multe astfel de zone. Cea mai recent creată şi încă neştearsă este considerată zonă curentă. Ori de cîte ori se rulează o comandă SQL, se adaugă timpul necesar execuţiei acesteia la timpul total, memorat în zona curentă de contorizare a timpului.

SHOW afişează titlul zonei curente de contorizare a timpului precum şi valoarea memorată în această zonă.

STOP afişează titlul şi valoarea zonei curente şi apoi şterge această zonă. Dacă mai există alte zone de contorizare a timpului cea mai recent creată devine zonă curentă.

Exemplu:

Să se creeze o zonă de contorizare a timpului necesar executării diferitelor comenzi, cu numele CONTOR.

```
SQL> TIMI START CONTOR
SQL> SELECT * FROM SALARIATI;
MARCA NUME      FUNCT  CODD SALA VENS CODS
-------------------------------------------------------------
1111`AVRAM ION   VINZATO 100000 21200 1000 1000
                          ...
4000 PAUL STEFAN SEF DEP 160000 35000 5600 7000
17 records selected.
elapsed seconds: 4
```

8.4. Stabilirea rîndurilor avansate pe următoarea pagină

Specificarea numărului de rînduri avansate la începutul paginii următoare se realizează cu ajutorul comenzii:

Scanned with OKEN Scanner

NEWPAGE [1 | n];

unde:
n desemnează numărul liniilor care vor avansa.

Exemplu:

Să se indice efectuarea unui salt de trei rînduri la începutul unei pagini noi.

```
SQL> SET PAGESIZE 7
SQL> TTITLE CENTER L I S T A   S A L A R I A T I L O R
SQL> NEWPAGE 3
SQL> RUN
  1  SELECT *
  2  FROM SALARIATI

     L I S T A   S A L A R I A T I L O R
MARCA NUME    FUNCT    CODD SALA VENS CODS
------------------------------------------------------------
1111 AVRAM ION   VINZATO 100000 21200 1000 1000
1222 BARBU DAN  VINZATO 120000 20750 2000 1000
2650 VLAD ION    VINZATO 120000 25060 3500 1000
2550 FRINCU ION  SEF DEP 160000 36000 37000 2500
1680 RADU ION    VINZATO 130000 20750 3000 2500


     L I S T A   S A L A R I A T I L O R
MARCA NUME    FUNCT    CODD SALA VENS CODS
------------------------------------------------------------



3500 DAN ION     VINZATO 160000 24500  350 2500
3700 MANU DAN    VINZATO 160000 27500 2500 3755
```


Scanned with OKEN Scanner

```
                LISTA SALARIATILOR
MARCA NUME    FUNCT    CODD SALA VENS CODS
-------------------------------------------------------------

3759 ALEXE IOAN       VINZATO  160000   25700
17 records selected.
```

8.5. Lansarea în execuţie a comenzilor SQL

Pentru lansarea unui fişier de comenzi se foloseşte următoarea sintaxă:

```
@nume-fişier[;]
```

Fişierul specificat poate conţine orice tip de comenzi SQL sau SQL*PLUS. El poate să conţină informaţii utile definirii modului de lucru, a formatului de afişare a diferitelor coloane etc. În general, fişierul are extensia SQL.

Dacă se introduce o comandă în buffer-ul SQL se declanşează automat şi lansarea în execuţie a acesteia. Indiferent de succesul sau insuccesul execuţiei comenzii, aceasta rămîne în buffer-ul curent şi se pot opera modificări asupra ei, prin intermediul comenzilor de editare.

Pentru lansarea în execuţie a comenzilor din buffer se utilizează comanda:

```
R[UN][;]
```

Această comandă va afişa comenzile înainte de lansarea lor în execuţie.

Pentru lansarea în execuţie a comenzilor care se găsesc în buffer-ul curent, fără afişarea acestora, se utilizează comanda:

Scanned with OKEN Scanner

/[;]

Acest mod de lansare se utilizează în fişiere de comenzi, cînd se doreşte ca pe ecran să apară numai rezultatele execuţiei comenzilor nu şi textul acestora (se pot încărca în buffer comenzile prin GET şi apoi lansa în execuţie cu /).

Pentru a lansa în execuţie un fişier care face apel la o serie de parametri, se dă comanda:

STA[RT] nume-fişier [param1 param2...][;]

Comanda va executa conţinutul fişierului specificat, în care se găsesc apeluri la parametri prin substituţii de forma &1, &2 etc.

Dacă nu se specifică explicit tipul fişierului, se va considera extensia **.SQL.**

Valorile sînt substituite numai pentru rularea în curs, astfel: prima valoare pentru &1, a doua valoare pentru &2 ş.a.m.d.

Exemple:

1) Să se lanseze în execuţie programul aflat în buffer-ul curent, utilizînd comanda **RUN.**

```
SQL> R
  1  SELECT *
  2* FROM SALARIATI;
MARCA NUME      FUNCT   CODD SALA VENS CODS
-------------------------------------------
1111 AVRAM ION    VINZATO 100000 21200 1000 1000
                    ...
4000 PAUL STEFAN SEF DEP 160000 35000 5600 7000
17 records selected.
```

2) Să se lanseze în execuţie programul aflat în fişierul de comenzi LUNGU.TXT.

Scanned with OKEN Scanner

```
SQL>@LUNGU.TXT
SELECT *
FROM SALARIATI
MARCA NUME    FUNCT   CODD SALA VENS CODS
------------------------------------------------------------------------
1111 AVRAM ION  VINZATO 100000 21200 1000 1000
1222 BARBU DAN  VINZATO 120000 20750 2000 1000
                                ...
4000 PAUL STEFAN SEF DEP 160000 35000 5600 7000
17 records selected.
```

3) Să se execute programul din buffer-ul curent şi să se indice valorile solicitate pentru parametri.

```
SQL> START TEXT.TXT &1
Enter value for 1: FUNCT
Enter value for 2: NUME
old   1: SELECT &1,&2
new   1: SELECT FUNCT,NUME
SQL>
  1  SELECT &1,&2
  2  FROM SALARIATI
  3  WHERE FUNCT ='DIRECTO'
FUNCT   NUME
------------------------
DIRECTO ION ION
```

8.6. Utilizarea variabilelor utilizator

Utilizatorul poate să-şi creeze propriile variabile, în scopul de a folosi, în comenzile SQL, valorile asignate acestora. Formulările de tipul **&nume_variabilă** din comenzile SQL vor fi substituite cu valorile atribuite variabilelor respective.

O variabilă utilizator se poate crea prin citirea unei valori introdusă de la tastatură, prin comanda **ACCEPT**, care are următoarea sintaxa:

Scanned with OKEN Scanner

ACC[EPT] nume-variabilă [NUM[BER] | CHAR] [PROMPT text | NOPR[OMPT] [HIDE];

Dacă variabila definită de **nume-variabilă** există, i se va atribui valoarea introdusă iar dacă nu există ea va fi creată.

NUMBER sau **CHAR** determină tipul variabilei (numeric, respectiv caracter). Dacă valoarea introdusă nu este conformă tipului se va genera un mesaj de eroare.

PROMPT specifică textul afişat ca prompter, înaintea citirii valorii de la tastatură; **NOPROMPT** determină aşteptarea introducerii fără a afişa prompter-ul.

HIDE inhibă vizualizarea valorii introduse.

Pentru definirea unei variabile utilizator şi atribuirea unei valori sau pentru afişarea valorii şi tipului unei variabile, se utilizează comanda **DEFINE**, care are două forme, corespunzătoare celor două scopuri enunţate.

Prima formă este:

DEF[INE] nume-variabilă = text;

Se defineşte o variabilă utilizator, atribuindu-i-se o valoare de tip şir de caractere.

A doua formă este:

DEF[INE] [nume-variabilă];

Se va afişa valoarea şi tipul variabilei specificate de **nume-variabilă.**

Dacă nu se specifică **nume-variabilă** se vor afişa valorile şi tipurile tuturor variabilelor definite de utilizator.

O variabilă poate fi definită şi prin introducerea ei de la tastatură conform variabilelor de substituţie declarate în program.

Pentru ştergerea definirii unei variabile utilizator se utilizează comanda:

Scanned with OKEN Scanner

UNDEF[INE] nume-variabilă;

Această comandă va şterge definirea variabilei specificate de nume-variabilă.

Exemple:

1) Să se definească variabila VAR1 ca fiind coloana MARCA, prin comanda **DEFINE** şi variabila VAR2 ca fiind coloana NUME, prin comanda **ACCEPT.**

```
SQL> DEF VAR1=MARCA;
SQL> ACC VAR2
NUME
SQL> /
old   1: SELECT &VAR1,&VAR2
new   1: SELECT MARCA,NUME
MARCA NUME
------------------------
7000 ION ION
```

2) Să se şteargă definirile VAR1 şi VAR2 din exemplul precedent şi să fie din nou definite prin introducerea de la tastatură.

```
SQL> UNDEF VAR1;
SQL> UNDEF VAR2;
SQL> /
Enter value for var1: MARCA
Enter value for var2: NUME
old   1: SELECT &VAR1,&VAR2
new   1: SELECT MARCA,NUME
 MARCA NUME
------------------------
  7000 ION ION
```


Scanned with OKEN Scanner

8.7. *Introducerea de comentarii printre comenzi*

Pentru introducerea de comentarii printre comenzile care se găsesc într-un fişier, există mai multe posibilităţi:

- utilizarea comenzii **REMARK;**
- utilizarea comenzii **DOCUMENT;**
- utilizarea caracterelor de separare **/* ...*/.**

Sintaxa comenzii **REM[ARK]** este:

REM[ARK] text-comentariu;

Ea specifică începerea unui comentariu care se va termina la sfirşitul liniei de comandă.

Sintaxa comenzii **DOC[UMENT]** este:

DOC[UMENET];

Ea specifică începutul unui bloc de documentare (comentariu) care se va termina prin introducerea caracterului #, singur pe o linie.

Cînd se execută un fişier de comenzi care conţine un bloc de documentare, acesta va fi afişat chiar dacă comanda **SET ECHO OFF** este activă. Singurul mod de a nu se afişa acest bloc este de a utiliza comanda **SET TERMOUT OFF.**

Cînd s-a dat comanda **SET DOCUMENT OFF** şi se execută un fişier de comenzi care are şi blocuri de documentare, comanda **DOCUMENT** va fi ignorată iar liniile întîlnite vor fi tratate ca linii de comandă.

Se mai pot introduce comentarii prin delimitarea acestora utilizînd perecheà de caractere **/*** şi ***/**. Textul ce se găseşte între cele două caractere-pereche va fi tratat drept comentariu.

Exemplu: Să se introducă un rînd de comentariu în programul din buffer-ul curent.

Scanned with OKEN Scanner

```
SQL> I/* PROGRAM DE TEST */
SQL> LIST
  1  SELECT &VAR1,&VAR2
  2  /* PROGRAM DE TEST */
  3  FROM SALARIATI
  4* WHERE FUNCT ='DIRECTO'
```

8.8 Introducerea de pauze printre comenzile SQL

Pentru ca după afişarea unei pagini să se facă pauză, urmînd să se aştepte apăsarea tastei **<Enter>**, se foloseşte comanda cu sintaxa:

```
PAU[SE] [text];
```

Această comandă are ca efect afişarea unei linii goale, urmată de o linie care conţine textul specificat, după care se aşteptă tastarea lui **<Enter>**.

Dacă nu se specifică textul, se vor afişa două linii goale.

Apăsarea tastei **<Enter>** este necesară chiar dacă sursa introducerii comenzilor a fost redirectată către un fişier de comenzi.

8.9. Afişarea unor informaţii

În vederea uşurării lucrului cu SQL*PLUS există comenzi care oferă informaţii în regim interactiv, fie despre modul în care au fost descrise tabelele la crearea lor, fie despre parametrii specifici diferitelor comenzi.

Pentru a obţine informaţii referitoare la descrierea tabelelor, fără a fi necesară consultarea cataloagelor de sistem, se utilizează comanda:

Scanned with OKEN Scanner

DESC[RIBE] [utilizator.] nume-tabelă;

Aceasta va afişa o scurtă descriere a tabelei specificate prin **nume-tabelă**, care poate să aparţină altui utilizator. În cazul în care nu se specifică utilizatorul, se consideră că el este proprietarul tabelei.

Descrierea afişată va conţine următoarele informaţii despre coloane: numele, dacă sînt permise valori nule sau nu, tipul, mărimea.

Pentru afişarea unor informaţii legate de comenzi se utilizează comanda **HELP**, care poate fi dată în următoarele forme:

HELP { comanda | clauza | cuvînt-cheie };
HELP { COMMANDS | CLAUSES };
HELP;

Prima formă va afişa informaţii legate de comanda, clauza sau cuvîntul cheie specificat.

HELP COMMANDS afişează o listă a comenzilor pentru care există informaţii disponibile; **HELP CLAUSES** afişează o listă a clauzelor pentru care se pot furniza informaţii;

HELP, singur, va afişa lista tuturor cuvintelor cheie.

Trebuie precizat că aceste informaţii se păstrează în baza de date, ocupînd un spaţiu destul de mare.

Din această cauză, pentru a elibera o parte din spaţiul ocupat din baza de date,se recomandă ca tabela care conţine informaţii de tip HELP să fie ştearsă (în acest caz la introducerea comenzii HELP se va obţine mesajul **Help is not available**, ca şi în cazul în care se doresc informaţii despre un cuvînt care nu este recunoscut de sistem).

Exemplu:
Să se afişeze informaţii referitoare la descrierea tabelei SALARIATI.

```
SQL> DESC SALARIATI
 Name                      Null?                    Type
 ------------------------------------------------------------------
 MARCA                     NOT NULL                 NUMBER(4)
```


Scanned with OKEN Scanner

NUME		CHAR(16)
FUNCT		CHAR(7)
CODD	NOT NULL	NUMBER(6)
SALA		NUMBER(9,2)
VENS		NUMBER(9,2)
CODS	NOT NULL	NUMBER(4)

8.10. Memorarea sesiunii de lucru într-un fişier

De multe ori este necesară memorarea sesiunii de lucru într-un fişier. Acest lucru se poate realiza cu ajutorul comenzii:

```
SPO[OL] [ nume-fişier | OFF | OUT ];
```

Prin specificarea numelui unui fişier, se comandă memorarea în acel fişier a tot ce se afişează pe ecran. Extensia implicită a fişierului este **.LST** sau **.LIS.**

OFF specifică oprirea acţiunii comenzii **SPOOL** iar **OUT** va avea efect asemănător, la care se adaugă tipărirea la imprimantă a fişierului. Dacă nu se specifică nici o clauză, se va afişa numele fişierului curent sau a celui mai recent folosit în comanda SPOOL.

Pentru a memora într-un fişier sesiunea de lucru, fără ca aceasta să apară pe ecran, se utilizează comanda **SET TERMOUT OFF.**

8.11. Lansarea de comenzi corespunzătoare sistemului de operare

SQL*PLUS permite lansarea de comenzi către sistemul de operare, fără să fie necesară întreruperea sesiunii de lucru. Acest lucru se realizează prin introducerea comenzii respective prefixată cu caracterul **$.**

De asemenea, se poate relansa fişierul **COMMAND.COM**, prin furnizarea doar a caracterului **$**, permiţînd astfel lucrul cu sistemul de

Scanned with OKEN Scanner

operare.

Pentru ieşirea din această stare şi reintrarea în SQL*PLUS se foloseşte comanda **EXIT**.

8.12. Permanentizarea sau renunţarea la modificările făcute asupra bazei de date

În vederea realizării fizice a modificărilor în baza de date se utilizează comanda:

```
COMMIT;
```

Comanda determină ca modificările făcute asupra tabelelor să devină permanente (să fie făcute fizic) iar blocările realizate prin comenzile **SELECT...FOR UPDATE** sau **LOCK TABLE** să fie eliberate.

Existenţa acestei comenzi va avea ca efect permanentizarea modificărilor făcute de următoarele comenzi: **ALTER, AUDIT, CONNECT, CREATE, DISCONNECT, DROP, EXIT, GRANT, NOAUDIT, QUIT, REVOKE, SAVE.**

De asemenea, prin comanda **SET AUTOCOMMIT ON** se poate specifica faptul că modificările se vor transmite în baza de date, în mod automat, la fiecare comandă dată.

În cazul în care se doreşte renunţarea la modificările făcute şi deci, netransmiterea lor asupra bazei de date, se poate utiliza comanda:

```
ROLLBACK;
```

Comanda va lua în considerare modificările realizate prin ultima comandă **COMMIT**, în condiţiile în care, în prealabil, s-a precizat comanda **SET AUTOCOMMIT OFF**. Un alt efect al comenzii ROLLBACK este şi acela de a elibera blocările existente.

În cazul în care o tranzacţie eşuează datorită unui incident (spre exemplu căderea tensiunii), SQLPLUS va executa, în mod automat, comanda **ROLLBACK**.

Se precizează faptul că după ce schimbările devin permanente, nu

Scanned with OKEN Scanner

se mai poate reveni şi restaura versiunea anterioară.

Exemple:

1) Să fie realizate efectiv modificările făcute asupra bazei.

```
SQL> COMMIT
commit complete
```

2) Să se renunţe la modificările făcute.

```
SQL> ROLLBACK;
rollback complete
```

8.13. Copierea datelor între baze de date diferite

Pentru copierea datelor dintr-o tabelă aflată într-o bază de date, într-o tabelă aflată în altă bază de date, se utilizează comanda:

```
COPY [FROM nume-utilizator/parolă@bază-date]
[TO nume-utilizator/parolă@bază-date]
{APPEND | CREATE | INSERT | REPLACE}
nume-tabelă [(nume-col, nume-col, ... )]
USING cerere;
```

unde:

FROM specifică numele utilizatorului, parola şi baza de date prin care se localizează tabela sursă. În mod similar, **TO** localizează tabela destinaţie. Dacă una din clauze este omisă, se va lua în considerare baza de date la care este conectat utilizatorul care a dat comanda. Se precizează faptul că sursa şi destinaţia nu pot avea aceeaşi bază de date, deci, din comanada **COPY**, nu pot lipsi ambele clauze.

APPEND, CREATE, INSERT, REPLACE vor determina modul de tratare a copiei în raport cu tabela de destinaţie.

APPEND specifică faptul că datele se vor adăuga la tabela existentă, iar în cazul în care tabela nu există aceasta va fi creată.

Scanned with OKEN Scanner

CREATE specifică faptul că tabela va trebui creată iar în cazul în care aceasta există se va genera un mesaj de eroare.

INSERT specifică faptul că datele se vor adăuga la tabela existentă, iar dacă tabela nu există se va furniza un mesaj de eroare.

REPLACE specifică faptul că datele vor înlocui pe cele anterioare din tabelă, iar dacă tabela nu există, va fi creată.

Nume-col specifică numele atribuit coloanelor în tabela de destinaţie. Dacă numele coloanelor tabelei de destinaţie sînt specificate, numărul acestora trebuie să fie egal cu cel al coloanelor care vor apare în cerere. Dacă numele coloanelor tabelei de destinaţie nu se specifică, atunci coloanele copiate vor avea acelaşi nume ca cele din tabela sursă.

Prin cerere se identifică tabela sursă şi se precizează care rînduri şi coloane vor fi copiate din aceasta.

Variabila de sistem **LONG** limitează lungimea unui cîmp de tip LONG care trebuie copiat. Dacă o coloană conţine cîmpuri mai lungi decît valoarea variabilei de sistem, aceste cîmpuri vor fi trunchiate în timpul copierii.

La sfîrşitul fiecărei copieri terminate cu succes, se va trimite automat o comandă **COMMIT**. Dacă variabila de sistem **COPY COMMIT** este setată la o valoare mai mare ca zero, comanda **COMMIT** se va trimite după ce s-a copiat un număr de pachete de înregistrări egal cu valoarea setată. Dimensiunea unui pachet este determinată de variabila de sistem **ARRAYSIZE**.

Scanned with OKEN Scanner

CAPITOLUL 9
FORMATAREA SITUAŢIILOR DE IEŞIRE

Limbajul SQL*PLUS permite proiectarea şi formatarea diverselor situaţii de ieşire. Aceste operaţii sînt posibile prin utilizarea unor comenzi pentru tratarea întreruperilor, comenzi pentru definirea titlurilor, definirea coloanelor şi stabilirea diverselor opţiuni pentru poziţionarea unor arii de lucru pe ecran.

9.1. Tratarea întreruperilor

Întreruperea este un eveniment care se produce în timpul execuţiei unei comenzi **SELECT**, cum ar fi, de exemplu, apariţia sfîrşitului de pagină sau schimbarea valorii unei expresii.

Pentru a specifica evenimentele care determină o întrerupere şi acţiunea corespunzătoare SQL care se execută, se utilizează comanda **BREAK**. Ea poate specifica mai multe evenimente care generează o întrerupere. Evenimentele sînt reţinute într-o ordine numită *"ierarhie de întrerupere"*. La un moment dat, se poate executa doar o singură comandă **BREAK**.

Comanda **BREAK** are următoarele forme sintactice:

```
BRE[AK] ON {expr | ROW | PAG[E] | REPORT}...
        [SKI[P] n | [SKIP]PAGE]
        [NODUP[LICATES] | DUP[LICATES]];
BRE[AK];
```

unde:

ON expr determină o întrerupere cînd se schimbă valoarea expresiei **expr**; **expr** este fie o expresie care implică una sau mai multe coloane dintr-o tabelă, fie o etichetă ataşată unei coloane declarată în comanda **SELECT** sau **COLUMN**.

Dacă **ON expr** apare de mai multe ori în comandă, atunci expresiile respectă "ierarhia de întrerupere", aceasta fiind chiar ordinea în care sînt specificate expresiile. Cînd se foloseşte **ON expr**, trebuie utilizată şi clauza **ORDER BY**, pentru a ordona rîndurile din

Scanned with OKEN Scanner

"ierarhia de întrerupere". Ordinea de apariţie a expresiilor expr în comanda **BREAK ON** trebuie să fie aceeaşi cu cea prezentă în **SELECT...ORDER BY;**

ON ROW determină o întrerupere cînd se selectează un rînd cu SELECT. Această întrerupere este reţinută la sfîrşitul ierarhiei de întrerupere;

ON PAGE determină o întrerupere la sfîrşitul fiecărei pagini;

ON REPORT determină o întrerupere la sfîrşitul raportului sau cererii, întrerupere care este reţinută la începutul ierarhiei de întrerupere.

SKIP determină saltul peste **n** linii, înainte de a se tipări linia asociată întreruperii respective.

PAGE sau SKIP PAGE determină saltul la o pagină nouă înainte de a tipări linia asociată respectivei întreruperi.

NODUPLICATES determină tipărirea de spaţii cînd valorile de pe coloana de întrerupere sînt identice. **DUPLICATES** determină tipărirea valoarii coloanei de întrerupere în fiecare rînd selectat. Valoarea NODUPLICATES este implicită.

Comanda **BREAK** fară clauze indică poziţia întreruperii curente.

Exemple:

1) Să fie definită o întrerupere generată de schimbarea valorilor coloanei **FUNCT**. La apariţia acestui eveniment să fie afişate două linii vide.

```
SQL> BREAK ON FUNCT SKIP 2;
SQL> SELECT MARCA,NUME,CODD,
  2 SALA,VENS
  3 FROM SALARIATI
  4 ORDER BY FUNCT;
MARCA NUME     FUNCT                CODD   SALA VENS
-----------------------------------------------------
7000 ION ION   DIRECTO              100000  45000 40000

2550 FRINCU ION  SEF DEP            160000  36000 37000
1000 COMAN RADU                     130000  35000  2500
```

2500 VLAD VASILE	160000	36500	1500
4000 PAUL STEFAN	160000	35000	5600
3755 DORU DAN	130000	36500	5500

1111 AVRAM ION VINZATO	100000	21200	1000
1680 RADU ION	130000	20750	3000
3700 MANU DAN	160000	27500	2500
2553 AILENEI FLORIN	120000	25000	400
3760 SANDU ION	130000	25600	0
3770 CARMEN ANA	130000	26500	
2554 DARIAN GEO	120000	26000	2000
3759 ALEXE IOAN	160000	25700	
3500 DAN ION	160000	24500	350
2650 VLAD ION	120000	25060	3500
1222 BARBU DAN	120000	20750	2000

2) Să fie definită o întrerupere la apariţia unei schimbări a valorii coloanei CODS din tabela SALARIATI. În momentul realizării evenimentului să se realizeze salt la pagină nouă.

```
SQL> SET PAGESIZE 11
SQL> BREAK ON CODS SKIP PAGE;
SQL> SELECT * FROM SALARIATI
  2 ORDER BY CODS;
```

MARCA NUME	FUNCT	CODD	SALA	VENS	CODS
1111 AVRAM ION	VINZATO	100000	21200	1000	1000
2650 VLAD ION	VINZATO	120000	25060	3500	
1222 BARBU DAN	VINZATO	120000	20750	2000	

MARCA NUME	FUNCT	CODD	SALA	VENS	CODS
2550 FRINCU ION	SEF DEP	160000	36000	37000	2500
3500 DAN ION	VINZATO	160000	24500	350	
1680 RADU ION	VINZATO	130000	20750	3000	

Scanned with OKEN Scanner

```
MARCA NUME      FUNCT          CODD SALA VENS CODS
-------------------------------------------------------------------------
2553 AILENEI FLOR VINZATO        120000 25000 400 2550
2554 DARIAN GEO   VINZATO        120000 26000 2000

MARCA NUME      FUNCT          CODD SALA VENS CODS
-------------------------------------------------------------------------
3700 MANU DAN    VINZATO         160000 27500 2500 3755
3759 ALEXE IOAN  VINZATO         160000 25700

MARCA NUME      FUNCT          CODD SALA VENS CODS
-------------------------------------------------------------------------
3760 SANDU ION    VINZATO        130000 25600    0 4000
3770 CARMEN ANA   VINZATO        130000 26500

MARCA NUME      FUNCT          CODD SALA VENS CODS
-------------------------------------------------------------------------
1000 COMAN RADU  SEF DEP         130000 35000 2500 7000
2500 VLAD VASILE SEF DEP         160000 36500 1500
4000 PAUL STEFAN SEF DEP         160000 35000 5600
3755 DORU DAN    SEF DEP         130000 36500 5500

MARCA NUME      FUNCT          CODD SALA VENS CODS
-------------------------------------------------------------------------
7000 ION ION     DIRECTO         100000 45000 40000 8000
17 records selected.
```

3) Să se indice întreruperea curentă.

```
SQL> BREAK;
break on FUNCT skip 2 dup
```

9.2. *Tipărirea titlurilor*

Pentru afişarea unui titlu la sfîrşitul sau la începutul fiecărei pagini se utilizează comanda **BTITLE**, respectiv **TTITLE**.

Comanda **BTITLE** are următoarele sintaxe:

Scanned with OKEN Scanner

```
BTI[TLE] [COL[UMN] n] [SKIP [n]] [TAB n]
        [LEFT | RIGHT | CENTER]
        [FORMAT char]
        [char | var] ... ;
BTI[TLE] {OFF | ON};
BTI[TLE] text;
BTI[TLE];
```

unde:

COL[UMN] n determină saltul la coloana **n** a liniei curente.

Prin **SKIP n** se determină saltul la începutul liniei curente, de **n** ori. Dacă **n** este omis, se sare o singură dată, iar dacă **n** este zero, se realizează întorcerea la începutul liniei curente.

TAB n are ca efect saltul a **n** coloane (prin coloană înţelegîndu-se nu o coloană de tabelă, ci poziţia cursorului) înainte dacă **n** este pozitiv sau înapoi dacă **n** este negativ.

Clauzele **LEFT, RIGHT, CENTER** determină alinierea la stînga, la dreapta respectiv afişarea centrată a datelor din linia curentă. Următoarele date sînt aliniate ca un grup, de la începutul pînă la sfîrşitul comenzii **PRINT** sau la următorul **LEFT, CENTER, RIGHT** sau **COLUMN. CENTER** şi **RIGHT** folosesc valoarea returnată de comanda **SET LINESIZE** pentru a calcula poziţia următorului articol.

FORMAT char specifică un model de format pentru articolul de date care urmează; acest format se menţine pînă la întîlnirea unei alte clauze **FORMAT** sau pînă la sfîrşitul comenzii. De reţinut că doar un singur model de format poate fi activ la un moment dat. Dacă nici un format potrivit nu are efect, atunci valorile sînt tipărite în conformitate cu formatul specificat în **SET NUMFORMAT** iar dacă **SET UNFORMAT** nu a fost utilizat, valorile vor fi tipărite cu formatul implicit.

Pentru specificarea titlurilor pot fi folosite constante (**char**) şi variabile (**var**), ele fiind poziţionate şi formatate aşa cum se specifică în clauzele comenzii.

Existenţa unui separator indică începutul unor linii noi, iar doi separatori pe acelaşi rînd introduc o linie vidă. Iniţial, caracterul de

Scanned with OKEN Scanner

separare este "|", însă el poate fi schimbat cu **SET HEADSEP**.

SQL*PLUS interpretează comanda BTITLE în forma nouă dacă primul cuvînt după numele comenzii este numele unei clauze valide (**LEFT, SKIP, COL** etc.).

Clauzele ON şi OFF au ca efect apariţia (ON) sau nu (OFF) a titlului.

BTITLE text afişează centrat textul specificat.

Comanda **BTITLE** fără clauze specifică titlul curent.

Exemple:

1) Să se afişeze la sfîrşitul unui raport final cu privire la produsele din depozitul cu codul 100000, în partea stîngă şirul de caractere "Data:" iar în partea dreaptă "Semnatura:".

```
SQL> SET PAGESIZE 11
SQL> BTITLE LEFT Data: RIGHT Semnatura:;
SQL> SELECT * FROM PRODUSE
  2  WHERE CODD=100000;
```

CODD	CODP	DENP	STOC	DATACRT	UM
100000	166666	PLACAJ 2/2	100	12-JUL-92	MP
100000	144444	SCAUN D4	36	12-JUL-92	BUC
100000	111111	MESE 15/20	7	27-JUN-92	BUC
100000	122222	FOTOLIU A3	12	01-JUL-92	BUC
100000	133333	CANAPEA A7	6	18-JUL-92	BUC
100000	155555	BIROU C6X4	9	29-JUL-92	BUC

Data: Semnatura:

2) Să se afişeze la sfîrşitul unui raport privind situaţia produselor din depozitul 100000, începînd din coloana 11, şirul de caractere "OBSERVATII".

```
SQL> BTITLE COL 11 OBSERVATII;
SQL> SELECT * FROM PRODUSE
  2  WHERE CODD=100000;
```


Scanned with OKEN Scanner

CODD	CODP	DENP	STOC	DATACRT	UM
100000	166666	PLACAJ 2/2	100	12-JUL-92	MP
100000	144444	SCAUN D4	36	12-JUL-92	BUC
100000	111111	MESE 15/20	7	27-JUN-92	BUC
100000	122222	FOTOLIU A3	12	01-JUL-92	BUC
100000	133333	CANAPEA A7	6	18-JUL-92	BUC
100000	155555	BIROU C6X4	9	29-JUL-92	BUC

OBSERVATII

3) Să se afişeze la sfîrşitul raportului privind produsele din depozitul cu codul 100000 următoarele şiruri de caractere "DATA" şi "DEPOZITUL MOBILA/100000". Afişarea să se facă pe două rînduri.

```
SQL> SET LINESIZE 11
SQL> SET HEADSEP |
SQL> BTITLE DATA|DEPOZITUL_MOBILA/100000
SQL> SELECT * FROM PRODUSE
  2 WHERE CODD=100000;
```

CODD	CODP	DENP	STOC	DATACRT	UM
100000	166666	PLACAJ 2/2	100	12-JUL-92	MP
100000	144444	SCAUN D4	36	12-JUL-92	BUC
100000	111111	MESE 15/20	7	27-JUN-92	BUC
100000	122222	FOTOLIU A3	12	01-JUL-92	BUC
100000	133333	CANAPEA A7	6	18-JUL-92	BUC
100000	155555	BIROU C6X4	9	29-JUL-92	BUC

DATA
DEPOZITUL_MOBILA/100000

4) Să se afişeze centrat, la sfîrşitul unui raport privind produsele din depozitul cu codul 100000, şirul de caractere "Depozitul_MOBILA/100000".

```
SQL> BTITLE CENTER Depozitul_MOBILA/100000
```


Scanned with OKEN Scanner

```
SQL> SELECT CODD "Cod depozit",
  2  DENP "Denumire",
  3  CODP "Cod produs",
  4  STOC "Stoc",DATACRT "Data",
  5  UM
  6  FROM PRODUSE
  7  WHERE CODD=100000;
```

Cod depozit	Denumire	Cod produs	Stoc	Data	UM
100000	PLACAJ 2/2	166666	100	12-JUL-92	MP
100000	SCAUN D4	144444	36	12-JUL-92	BUC
100000	MESE 15/20	111111	7	27-JUN-92	BUC
100000	FOTOLIU A3	122222	12	01-JUL-92	BUC
100000	CANAPEA A7	133333	6	18-JUL-92	BUC
100000	BIROU C6X4	155555	9	29-JUL-92	BUC

Depozitul_MOBILA/100000

5) Să se afişeze specificaţiile curente pentru BTITLE. Ultima comandă primită în sistem se prezintă astfel:

```
SQL> BTITLE COL 40 Total RIGHT Semnatura;
SQL> BTITLE;
btitle ON and is the following 28 characters:
COL 40 Total RIGHT Semnatura
```

Comanda **TTITLE** determină afişarea unui titlu la începutul paginii şi are următoarele sintaxe:

```
TTI[TLE] [COL[UMN] n] [SKIP [1|n] [TAB n]
      [LEFT | RIGHT | CENTER]
      [FORMAT char] [char | var];
TTI[TLE] text;
TTI[TLE];
```

Clauzele comenzii TTITLE au aceeaşi semnificaţie ca la BTITLE.

Scanned with OKEN Scanner

Comanda TTITLE fără clauze determină afişarea titlului curent.

Exemple:

1) Să se afişeze la începutul unui raport privind comenzile cu data mai mică de 30 iulie 1992,următoarele şiruri de caractere: "S I T U A T I A C O M E N Z I L O R" şi "L A D A T A D E 30-JUL-92". Cele două şiruri se vor afişa centrat, pe două linii.

```
SQL> SET PAGESIZE 11
SQL> SET LIN 45
SQL> COLUMN NRCOM FORMAT 99999999
SQL> COLUMN NRCOM JUSTIFY CENTER
SQL> COLUMN CODP  FORMAT 99999999
SQL> COLUMN CODP JUSTIFY CENTER
SQL> COLUMN VALOARE FORMAT 99999999999
SQL> COLUMN VALOARE JUSTIFY CENTER
SQL> SET SPACE 7
SQL> TTITLE "S I T U A T I A C O M E N Z I L O R|L A
D A T A D E 30-JUL-92"
SQL> SELECT
  2 NRCOM,CODP,CANT*PRET VALOARE
  3 FROM COMENZI
  4 WHERE DATAL<='30-JUL-92';
Thu Jan 7                    page 1
   S I T U A T I A C O M E N Z I L O R
     L A D A T A D E 30-JUL-92

 NRCOM        CODP        VALOARE
-----------------------------------------------
 211111       133333      320000
 233333       144444       27000
 244444       166666      375000
 255566       222222      282000
```

2) Să se afişeze specificaţiile pentru titlul curent.

```
SQL> TTITLE;
```


Scanned with OKEN Scanner

ttitle ON and is the following 68 characters:
S I T U A T I A C O M E N Z I L O R L A D A T A D E
30-JUL-92

9.3. Anularea opţiunilor

În vederea anulării opţiunilor se utilizează comanda:

CL[EAR] option;

Este anulată opţiunea specificată prin **option**, după cum urmează:
BRE[AKS] anulează întreruperea indicată prin comanda **BREAK**;
BUFF[ER] determină ştergerea textului din buffer-ul curent;
COL[UMNS] anulează opţiunile indicate în comanda COLUMN;
COMP[UTES] anulează opţiunile indicate prin comanda COMPUTE;
SCR[EEN] determină ştergerea ecranului, iar SQL determină ştergerea buffer-ului SQL;
TIMI[NG] şterge toate zonele create cu comanda TIMING.

Exemple:

1) Să se anuleze întreruperile.

SQL> **CELAR BREAKS;**

2) Să se anuleze definiţiile de coloană.

SQL> **CLEAR COLUMNS;**

9.4. Descrierea coloanelor

Pentru specificarea modului de formare a unei coloane şi a capului de coloană într-o situaţie, se utilizează comanda **COLUMN**.
Sintaxa ei este:

Scanned with OKEN Scanner

```
COL[UMN] {col | expr}
  [ALI[AS] sinonim]
  [CLE[AR] | DEF[AULT]]
  [COLOR {culoare | variabilă-culoare}]
  [FOR[MAT] format]
  [HEA[DING] text
  [JUS[TIFY] {L[EFT | C[ENTER] | R[IGHT]]
  [LIKE {expr|etichetă}]
  [LINEAPP {LINE | MARK | BOTH}]
  [NEWL[INE]]
  [NEW_VALUE variabilă]
  [NU[LL] char]
  [NOPRI[NT] | PRINT]]
  [OLD_VALUE variabilă]
  [ON | OFF]
  [PATTERN {număr-de-model | variabilă-model}
  [WRA[PPED]|WOR[D_WRAPPED]|
          TRU[NCATED]]
  ...;
```

unde:

col sau **expr** au rolul de a identifica coloana sau expresia la care se referă comanda. Comanda COLUMN trebuie să se refere la o coloană sau la o expresie utilizată în comanda **SELECT**. Dacă comanda COLUMN se referă la o expresie, aceasta trebuie să fie specificată în acelaşi mod în care a fost specificată în SELECT. De exemplu, dacă în SELECT expresia este 'A+B', nu se poate folosi, în COLUMN, 'B+A'. Dacă sînt selectate din tabele diferite coloane cu acelaşi nume, atunci comanda **COLUMN** se va aplica tuturor tabelelor ce conţin acea coloană.

Clauza ALIAS pune în corespondenţă numele coloanelor cu sinonimul specificat.

Clauza CLEAR are ca efect ştergerea în totalitate a definiţiei coloanei.

Scanned with OKEN Scanner

Clauza **FORMAT** specifică formatul de apariţie al coloanei. Lăţimea implicită a coloanei este chiar lăţimea coloanei definite în baza de date. Ea poate fi schimbată cu valoarea **n**, folosind "**FORMAT A n** ". Lăţimea unei coloane numerice este implicit valoarea data de **NUMWIDTH**, ea putînd fi schimbată cu ajutorul clauzei **FORMAT**.

Clauza **HEADING** defineşte capul coloanei, care implicit este **col** sau **expr**. Dacă textul din definirea coloanei conţine spaţii sau semne de punctuaţie, acestea trebuie puse între ghilimele. Fiecare semn "|" din text are ca efect începerea unei linii noi.

Clauza **JUSTIFY** aliniază capul coloanei. Implicit, alinierea se face la dreapta pentru o coloană numerică şi la stînga pentru alte tipuri de coloane.

Clauza **LIKE** determină copierea specificaţiilor altei coloane sau expresii (specificaţii deja definite prin altă comandă COLUMN).

Prin folosirea clauzei **NEWLINE** se trece la o linie nouă, înainte de a afişa valorile coloanei.

Clauzele **PRINT** şi **NOPRINT** au ca efect tipărirea sau nu a coloanei.

Dacă se utilizează clauza **NULL**, textul va fi afişat chiar dacă conţine o valoare nulă. Valoarea implicită este un şir de blancuri.

Clauzele **OFF** sau **ON** determină folosirea, respectiv nefolosirea formatului specificaţiei de ieşire pentru o coloană, fără a-i afecta conţinutul.

Clauza **WRAPPED** are ca efect scrierea pe linia următoare a caracterelor care nu au încăput pe prima linie.

Clauza **WORD_WRAPPED** are efect asemănător cu cel al clauzei **WRAPPED**, cu deosebirea că determină scrierea întregului cuvînt pe linia următoare şi nu pe două rînduri ca la WRAPPED.

Clauza **TRUNC** determină trunchierea valorii.

Clauza **COLOR** specifică culoarea utilizată pentru afişarea valorilor coloanei, într-o reprezentare liniară sau prin benzi. Variabila **CLR n** (n=1,60) se referă la valoarea curentă a culorii. Setarea se face prin comanda **SET CLR n**.

Clauza **LINEAPP** stabileşte modul de subliniere a coloanei. Utilizînd **LINE** dreapta apare continuă. Folosind **MARK** se schiţează doar puncte, iar **BOTH** realizează ambele modalităţi: linie şi puncte.

Clauza **PATTERN** specifică modelul pe care-l va avea dreptunghiul/banda într-un grafic de tip benzi sau sectorul într-un grafic de tip cerc. **număr-de-model** este cuprins între 1 şi 16 şi

Scanned with OKEN Scanner

reprezintă un model. Variabila **PAT n**, unde nε[1,30] se referă la poziţia curentă a modelului. Semnificaţia valorilor din număr-de-model şi a valorilor **PAT n** sînt precizate în documentaţie.

Numărul de comenzi **COLUMN** folosite indică faptul că se lucrează cu diverse coloane sau expresii utilizate în comanda **SELECT**. Pentru o coloană sau expresie pot exista mai multe comenzi **COLUMN**. În cazul în care mai multe comenzi **COLUMN** folosesc aceeaşi clauză pentru aceeaşi coloană, se va lua în considerare doar ultima comandă.

Comanda COLUMN fără clauze are ca efect tipărirea definiţiilor coloanei curente. Dacă această comandă are doar clauzele **col** şi **expr**, efectul ei constă în indicarea definirii coloanei existente.

Trebuie precizat că în prima comanda COLUMN, expresia trebuie să aibă aceeaşi formă ca şi în SELECT. În caz contrar, SQL*PLUS nu va putea executa comanda COLUMN pentru coloana respectivă.

Exemple:

1) Să se definească coloana NUME pe o lăţime de 25 de caractere iar capul de coloană să fie scris pe două rînduri, astfel:

```
NUME
PRODUS
SQL> COLUMN NUME FORMAT A25 HEADING 'NUME|
PRODUS';
SQL> SELECT DISTINCT(DENP) NUME
  2 FROM PRODUSE;

NUME
PRODUSE
-------------------
MESE 15/20
...
6 records selected.
```

2) Să se afişeze sinonimul SALARIU pentru coloana definită de expresie aritmetică SALA+VENS. Coloana va fi scrisă cu formatul $99,999.99.

```
SQL> COLUMN SALA+VENS ALIAS SALARIU
```


Scanned with OKEN Scanner

```
SQL> COLUMN SALARIU FORMAT $99,999.99
SQL> SELECT SALA+VENS SALARIU,
  2  MARCA, NUME
  3  FROM SALARIATI;
   SALARIU          MARCA    NUME
-----------------------------------------------
$22,200.00          1111 AVRAM ION
$22,750.00          1222 BARBU DAN
                      ...
16 records selected.
```

3) Să se definească coloana DENP de 16 caractere alfanumerice, pentru care se specifică sinonimul DENUMIRE . În capul de coloană se va afişa textul "Denumire produs".

```
SQL> COLUMN DENP FORMAT A16
SQL> COLUMN DENP ALIAS DENUMIRE
SQL> COLUMN DENUMIRE HEADING "Denumire produs"
SQL> SELECT * FROM PRODUSE;
CODD CODP Denumire produs          STOC DATACRT  UM
-------------------------------------------------------------------
100000 166666 PLACAJ 2/2           100 12-JUL-92 MP
100000 144444 SCAUN D4              36 12-JUL-92 BUC
100000 111111 MESE 15/20             7 27-JUN-92 BUC
100000 122222 FOTOLIU A3            12 01-JUL-92 BUC
100000 133333 CANAPEA A7             6 18-JUL-92 BUC
100000 155555 BIROU C6X4             9 29-JUL-92 BUC
6 records selected.
```

4) Să se definească coloana DENUM2 cu un format alfanumeric de trei caractere. Specificaţiile pentru această coloană sînt copiate pentru coloana DENP.

```
SQL> COLUMN DENUM2 FORMAT A3
SQL> COLUMN DENP LIKE DENUM2
SQL> SELECT * FROM PRODUSE
  2  WHERE CODP<133333;
```


Scanned with OKEN Scanner

```
CODD   CODP DEN    STOC DATACRT   UM
------------------------------------------------------------
100000 111111 MES      7 27-JUN-92 BUC
              E 1
              5/2
              0
100000 122222 FOT     12 01-JUL-92 BUC
              OLI
              U A
              3
```

5) Să se definească coloana ADRESA pe 21 de caractere, utilizînd clauza WRAPPED.

```
SQL> COLUMN FORMAT A21 ADRESA WRAPPED
SQL> SELECT STR||'-'||NR||'-'||LOC ADRESA
  2  FROM CLIENTI;

ADRESA
---------------------
MOSILOR-104-BUCUREST
I
DOROBANTI-18-BUCURES
TI
COLENTINA-221-BUCURE
STI
EMINESCU-44-BUCUREST
I
4 records selected.
```

6) Să se definească coloana ADRESA pe 21 de caractere, utilizînd clauza TRUNC.

```
SQL> COLUMN ADRESA TRUN
SQL> SELECT STR||'-'||NR||'-'||LOC ADRESA
  2  FROM CLIENTI;
ADRESA
---------------------
MOSILOR-104-BUCUREST
DOROBANTI-18-BUCURES
```


Scanned with OKEN Scanner

COLENTINA-221-BUCURE
EMINESCU-44-BUCUREST

7) Să se definească coloana NUME, utilizînd clauzele WOR şi HEADING.

```
SQL> COLUMN NUME FORMAT A7 WOR
SQL> COLUMN NUME HEADING 'NUME|PRENUME'
SQL> SELECT MARCA,NUME,CODD,CODS
  2  FROM SALARIATI;
```

MARCA	NUME PRENUME	CODD	CODS
1111	AVRAM ION	100000	1000
1222	BARBU DAN	120000	1000
	....		

16 records selected.

8) Să se afişeze pe ecran coloana SALARIU, în culoarea albastru şi cu modelul 3.

```
SQL> COLUMN SALARIU COLOR BLUE PATERN 3;
```

9) Să se afişeze pe ecran coloana SAL, în culoarea dată de valoarea curentă a variabilei CLR4 şi cu modelul specificat prin valoarea curentă a variabilei PAT1.

```
SQL> COLUMN SAL COLOR CLR4 PATTERN PAT1;
```

9.5. *Realizarea de calcule*

SQLPLUS permite realizarea unor calcule cu rîndurile selectate din tabele. Pentru aceasta se utilizează comanda COMPUTE, care are următoarea sintaxă:

```
COMP[UTE] [AVG|COU[NT]|MAX[IMUM]|
```


Scanned with OKEN Scanner

```
MIN[IMUM]|NUM[BER]|SDT|SUM|VAR[IANCE]]
...
OF {expresie|etichetă}, ...
ON [expresie|etichetă|PAGE|REPORT|ROW];
```

Semnificaţia clauzelor este prezentată în tabelul 9.1.

Tabelul 9.1.

Clauza	*Semnificaţia*	*Tipul coloanei la care se poate aplica*
AVG	Valoare medie	Numeric
COUNT	Contorizare valori nule	Toate tipurile
MAXIMUM	Valoare maximă	Numeric şi caracter
MINIMUM	Valoare minimă	Numeric şi caracter
NUMBER	Contorizare rînduri	Toate tipurile
STD	Abatere standard	Numeric
SUM	Suma valorilor nenule	Numeric
VARIANCE	Dispersia	Numeric

Dacă se specifică mai multe funcţii, nu trebuie separate între ele prin virgulă.

OF precizează coloana sau expresia ale căror valori se vor supune calculului. Acestea trebuie să apară în comanda SELECT, altfel comanda COMPUTE le va ignora. Dacă se doreşte ca valoarea calculată să nu fie afişată pe ecran se va utiliza clauza NON PRINT ataşată coloanei respective cu comanda SELECT.

Scanned with OKEN Scanner

ON specifică un element care va fi utilizat ca generator de întrerupere. El poate fi: expresie. etichetă, pagină (PAGE), raport (REPORT), linie (ROW). De cîte ori se va schimba valoarea acestuia, se va declanşa automat recalcularea funcţiei definite de COMPUTE.

În cazul în care se dă comanda fără nici o funcţie, se vor afişa specificaţiile de calcul definite.

Secvenţa de instrucţiuni care se va utiliza, în general, va fi:

SQL> BREAK ON expresie1

SQL> COMPUTE clauza OF expresie2 ON exprésie1;

SQL> SELECT expresie2,...;

Exemple:

1) Să se editeze situaţia finală cu următorul format:

S I T U A T I E

SUM(SALA) FUNCTIE

.....

Data: **Semnatura:**

```
SQL> SET PAGESIZE 22
SQL> RUN
  1  SELECT SUM(SALA),FUNCT FROM SALARIATI
  2* GROUP BY FUNCT

                    S I T U A T I E
SUM(SALA) FUNCT
----------------------------
    45000 DIRECTO
   179000 SEF DEP
   268560 VINZATO
```

Data: Semnatura:

Scanned with OKEN Scanner

2) Să se editeze o situaţie finală în care se separă coloana SALA în trei coloane corespunzătoare celor trei depozite.

```
SQL> SET PAGESIZE 24
SQL> RUN
  1  SELECT FUNCT,NUME,
  2  DECODE(CODD,100000,SALA,0) "DEP 100000",
  3  DECODE(CODD,130000,SALA,0) "DEP 130000",
  4  DECODE(CODD,160000,SALA,0) "DEP 160000"
  5* FROM SALARIATI
```

S I T U A T I E

FUNCT	NUME	DEP 100000	DEP 130000	DEP 160000
VINZATO	AVRAM ION	21200	0	0
	BARBU DAN	0	0	0
	VLAD ION	0	0	0
SEF DEP	FRINCU ION	0	0	36000
VINZATO	RADU ION	0	20750	0
	DAN ION	0	0	24500
	MANU DAN	0	0	27500
	ALEXE IOAN	0	0	25700
	AILENEI FLORIN	0	0	0
	DARIAN GEO	0	0	0
	SANDU ION	0	25600	0
	CARMEN ANA	0	26500	0
DIRECTO	ION' ION	45000	0	0
SEF DEP	COMAN RADU	0	35000	0
	VLAD VASILE	0	0	36500
	DORU DAN	0	36500	0
	PAUL STEFAN	0	0	35000

Data: Semnatura:

3) Să se editeze o situaţie finală cu salariile grupate pe funcţii şi depozite.

```
SQL> RUN
1 SELECT FUNCT,
2 SUM(DECODE(CODD,100000,SALA,0)) "DEP 10",
3 SUM(DECODE(CODD,130000,SALA,0)) "DEP 13",
4 SUM(DECODE(CODD,160000,SALA,0)) "DEP 16"
```


Scanned with OKEN Scanner

```
5 FROM SALARIATI
6 GROUP BY FUNCT;

                S I T U A T I E

FUNCT       DEP 10      DEP 13      DEP 16
------- ---------- ---------- ----------
DIRECTO      45000           0           0
SEF DEP          0       71500      107500
VINZATO      21200       72850       77700

Data:                                        Semnatura:
```

4) Să se afişeze o situaţie finală cu salariile grupate pe funcţii şi depozite. De asemenea, să se introducă o coloană cu suma salariilor pe fiecare funcţie.

```
SQL> RUN
1 SELECT FUNCT,
2 SUM(DECODE(CODD,100000,SALA,0)) "DEP 10",
3 SUM(DECODE(CODD,130000,SALA,0)) "DEP 13",
4 SUM(DECODE(CODD,160000,SALA,0)) "DEP 16"
5 ,SUM(SALA)
6 FROM SALARIATI
7 GROUP BY FUNCT;

                   S I T U A T I E

FUNCT       DEP 10      DEP 13      DEP 16 SUM(SALA)
------- ---------- ---------- ---------- ----------
DIRECTO      45000           0           0      45000
SEF DEP          0       71500      107500     179000
VINZATO      21200       72850       77700     268560

Data:                                        Semnatura:
```

5) Să se afişeze o situaţie finală cu salariile grupate pe funcţii şi depozite. De asemenea, să se facă totalul salariilor pe fiecare depozit şi fiecare funcţie. Forma situaţiei finale să se prezinte astfel:

Scanned with OKEN Scanner

```
S I T U A T I A   S A L A R I A T I L O R
P E   D E P O Z I T E   /   F U N C T I I

FUNCT      DEP 10      DEP 13      DEP 16    SUMA
----- ----------- ----------- ----------- ------

                  ............

      ----------- ----------- ----------- ------
TOTAL: _________   _________   _________

Data:                                    Semnatura:

SQL> BREAK ON DUMMY;
SQL> COMPUTE SUM OF "DEP 100000" ON DUMMY;
SQL> COMPUTE SUM OF "DEP 130000" ON DUMMY;
SQL> COMPUTE SUM OF "DEP 160000" ON DUMMY;
SQL> COMPUTE SUM OF "TOTAL" ON DUMMY;
SQL> COLUMN DUMMY NOPRINT;
SQL> TTITLE CENTER "S I T U A T I A   S A L A R I I L O
R" SKIP CENTER "P E   D E P O Z I T E   /   F U N C T I I"
SKIP CENTER " "
SQL> RUN
1 SELECT FUNCT,
2 SUM(DECODE(CODD,100000,SALA,0)) "DEP 10",
3 SUM(DECODE(CODD,130000,SALA,0)) "DEP 13",
4 SUM(DECODE(CODD,160000,SALA,0)) "DEP 16"
5  ,SUM(SALA)
6  ,SUM(0) DUMMY
7 FROM SALARIATI
8 GROUP BY FUNCT
```

S I T U A T I A S A L A R I I L O R
P E D E P O Z I T E / F U N C T I I

FUNCT	DEP 10	DEP 13	DEP 16	SUM(SALA)
DIRECTO	45000	0	0	45000
SEF DEP	0	71500	107500	179000
VINZATO	21200	72850	77700	268560
	66200	144350	185200	

Data:

Semnatura:

Scanned with OKEN Scanner

CAPITOLUL 10
COMENZI PENTRU LUCRUL ÎN REŢEA

Sistemul de gestiune a bazelor de date ORACLE permite atît lucrul cu o bază de date locală cît şi lucrul în reţea. Limbajul SQL*PLUS folosit pentru lucrul în reţea nu diferă de cel utilizat pentru lucrul cu o bază de date locală.

Pentru a putea lucra cu o bază de date aflată la distanţă trebuie mai întîi să se creeze o legătură între cele două baze de date, tabelele aflate în baza de date de la distanţă fiind referite şi prin intermediul legăturii.

10.1. Crearea legăturii cu baza de date de la distanţă

Prin crearea unei legături între baza de date locală şi o bază de date aflată la distanţă se utilizează comanda CREATE DATABASE LINK cu următoarea sintaxă:

```
CREATE [PUBLIC] DATABASE
       LINK nume-legătură
       CONNECT TO nume-utilizator
       IDENTIFIED BY parolă
       USING 'bază-de-date-la-distanţă';
```

Odată ce o legatură la baza de date a fost creată, în continuare, se pot face cereri asupra tabelelor corespunzătoare bazei de date de la distanţă, prin specificarea numelui legăturii în clauza FROM din cerere sau subcerere. Cererea sau subcererea se desfăşoară ca şi cum utilizatorul este conectat pur şi simplu la baza de date de la distanţă.

Dacă se omite PUBLIC, legătura este privată, ea putînd fi folosită numai de cel care a creat-o. Dacă opţiunea PUBLIC se specifică, aceasta înseamnă că legătura este disponibilă tuturor utilizatorilor cu excepţia celor care şi-au definit o legatură privată cu acelaşi nume. Utilizarea acestei clauze se poate face numai de cei cu drepturi de administrare a bazei de date.

Scanned with OKEN Scanner

Numele utilizatorului şi parola trebuie să fie valide pentru baza de date de la distanţă.

10.2. Ştergerea legăturii cu baza de date de la distanţă

Pentru ştergerea unei legături între două baze de date, una locală şi una la distanţă se utilizează comanda **DROP DATABASE LINK** cu următoarea sintaxă:

```
DROP [PUBLIC] DATABASE LINK nume-leg;
```

Dacă legătura a fost creată cu opţiunea **PUBLIC**, atunci ea trebuie specificată şi la ştergerea legăturii. Ca şi la crearea **PUBLIC** se poate da numai de cei cu drepturi de administrare.

Dacă legatura este privată, trebuie omisă opţiunea **PUBLIC**.

Scanned with OKEN Scanner

ANEXA 1
CUVINTE REZERVATE

ACCES
ADD
ALL
ALTER
AND
ANY
APPEND
AS
ASC
ASSERT
ASSIGN
AUDIT
BETWEEN
BY
CHAR
CHECK
CLUSTER
COLUMN
COMMENT
COMPRESS
CONNECT
CONTAIN
CONTAINS
CRASH
CREATE
CURRENT
DATAPAGES
DATE
DBA
DBLINK
DECIMAL
DEFAULT
DEFINITION
DELETE
DESC
DISTINCT
DOES
DROP
EACH
ELSE
ERASE
EVALUATE
EXCLUSIVE
EXISTS
FILE
FLOAT
FOR
FORMAT
FROM
GRANT
GRAPHIC
GROUP
HAVING
IDENTIFIED
IF
IMAGE
IMMEDIATE
IN
INCREMENT
INDEX
INDEXED
INDEXPAGES
INITIAL
INSERT
INTEGER
INTERSECT
INTO
IS
LEVEL
LIKE

Scanned with OKEN Scanner

LINK
LIST
LOCK
LONG
MAXEXTENTS
MINUS
MODE
MODIFY
MOVE
NEW
NOAUDIT
NOCOMPRESS
NOLIST
NOSYSSORT
NOT
NOWAIT
NULL
NUMBER
OF
OFFLINE
OLD
ON
ONLINE
OPTIMIZE
OPTION
OR
ORDER
PAGE
PARTITION
PCTFREE
PRIOR
PRIVILEGES
PUBLIC
RAW
RENAME
REPLACE
REPORT
RESOURCE
REVOKE
ROW
ROWID
ROWNUM
ROWS
RUN
SELECT
SESSION
SET
SHARE
SIZE
SMALLINT
SPACE
START
SUCCESSFUL
SYNONYM
SYSDATE
SYSSORT
TABLE
TEMPORARY
THEN
TO
TRIGGER
UID
UNION
UNIQUE
UPDATE
USER
USING
VALIDATE
VALUES
VARCHAR
VARGRAPHIC
VIEW
WHENEVER
WHERE
WITH

Scanned with OKEN Scanner

ANEXA 2
OPERATORI UTILIZAŢI ÎN LIMBAJUL SQL*PLUS

A. Sintaxa operatorilor SQL*PLUS

Operator	Funcţie
&	Specifică înlocuirile lexicale într-un fişier de comenzi executat cu START. Opţiunile sînt înlocuite prin &s: prima prin &1, a doua prin &2 etc.
&,&&	Indică o variabilă utilizator într-o comandă SQL*PLUS. Se cere o valoare de fiecare dată cînd variabila & este găsită şi se cere o valoare cînd se găseşte pentru prima dată variabila '&&'. Încheie o variabilă de substituţie urmată de un caracter ce ar putea fi parte a numelui variabilei.

B. Sintaxa operatorilor SQL

Operator	Funcţie
(...)	Include o subcerere conţinută într-o altă comandă.
'...'	Delimitează o constantă de tip caracter sau dată calendaristică. Un apostrof într-o constantă de tip caracter se reprezintă prin două apostrofuri.
"..."	Marchează un nume de coloană sau sinonim care conţine caractere speciale. Marchează literalii într-un format de tip dată calendaristică.
@	Precede un nume de legatură la o bază de date, într-o clauză FROM.

C. Operatorii aritmetici SQL

Operator	Funcţie
+ -	Operatorii unari + şi - (valori pozitive, respectiv negative).
* /	Operatori de înmulţire şi împărţire.

Scanned with OKEN Scanner

+ -	Operatori de adunare şi scădere.
| |	Concatenare de şiruri de caractere.

D. Operatorii logici

Operator	Funcţie
=	Egalitate.
!= ^= <>	Inegalitate.
> >= < <=	Mai mare, mai mare sau egal, mai mic, mai mic sau egal.
NOT IN	Diferit de orice valoare din mulţimea de valori ce urmează. Este echivalent cu != ALL (nu aparţine unei mulţimi).
IN	Egal cu o valoare oarecare din mulţimea de valori ce urmează. Este echivalent cu = ANY (aparţine unei mulţimi).
ANY	Indică o valoare oarecare dintr-o mulţime.
ALL	Indică toate valorile unei mulţimi.
[NOT] BETWEEN x AND y	Valoarea unei variabile nu se găseşte/se găseşte în intervalul [x,y].
EXISTS	Condiţia este adevarată dacă o subcerere returnează cel puţin un rînd.
[NOT] LIKE	Compară valoarea unui cîmp cu un şir de caractere. Dacă în şirul de caractere urmează: % se compară cu orice secvenţă de caractere; _ se compară cu orice caracter.
IS[NOT] NULL	Operatorul specifică dacă valoarea unei variabile este sau nu nulă.
NOT	Funcţia logică "NU".
AND	Funcţia logică "ŞI".
OR	Funcţia logică "SAU".

Scanned with OKEN Scanner

E. Operatori utilizaţi în expresiile de cereri

Operator	Funcţie
UNION	Combină mai multe cereri, returnînd reunirea liniilor selectate de cererile individuale.
INTERSECT	Combină mai multe cereri, returnînd intersecţia liniilor selectate de cererile individuale.
MINUS	Combină mai multe cereri, returnînd diferenţa liniilor selectate de două cereri (rîndurile distincte selectate de prima cerere şi neselectate de a doua cerere).

F. Alţi operatori SQL

Operator	Funcţie
(+)	Indică faptul că o coloană ce îl precede este coloană de joncţiune externă(OUTER JOIN COLUMN).
PRIOR	Defineşte relaţiile părinte-fiu între nodurile unei cereri structurate arborescent. Dacă operatorul PRIOR precede o expresie din stînga unei egalităţi, se face o selecţie descendentă. Dacă operatorul PRIOR precede o expresie din dreapta unei egalităţi se va face o selecţie ascendentă.

Scanned with OKEN Scanner

ANEXA 3
FUNCŢII, VARIABILE SISTEM ŞI PSEUDO-COLOANE

A. Funcţii aritmetice

Funcţii	Valoare returnată
ABS(n)	Valoarea absolută a lui **n**.
CEIL(n)	Cel mai mic întreg mai mare sau egal cu **n**.
FLOOR(n)	Cel mai mare întreg mai mic sau egal cu **n**.
MOD(m,n)	Restul împărţirii lui **m** la **n**.
POWER(m,n)	**m** la puterea **n**; **n** trebuie să fie întreg.
ROUND(n[,m])	**n** rotunjit la **m** zecimale; dacă **m** este omis se rotunjeşte la zero zecimale.
SIGN(n)	Semnul lui **n**. dacă **n** < 0, atunci **SIGN (n) = -1**. dacă **n** = 0, atunci **SIGN (n) = 0**. dacă **n** > 0, atunci **SIGN (n) = 1**.
SQRT(n)	Rădăcina pătrată din **n**. Dacă **n** < 0 atunci rezultatul este valoarea NULL.
TRUNC(n[m])	**n** trunchiat la **m** poziţii zecimale. Dacă **m** este omis trunchierea se face la zero poziţii zecimale.

B. Funcţii caracter

Funcţii	Valoare returnată
ASCII(char)	Se returnează valoarea internă ASCII a primului caracter din **char**.
CHR(n)	Se returnează caracterul cu valoarea internă ASCII **n**.
INITCAP(char)	Se returnează un şir de caractere **char** cu prima literă a fiecărui cuvînt transformată în literă mare.

Scanned with OKEN Scanner

INSTR(char-1,char-2[,n[,m]]) Se returnează poziţia celei de-a **m**-a apariţii a şirului de caractere **char-2** în şirul **char-1**; căutarea începe de pe poziţia **n**-a.Dacă **m** sau **n** sînt omise,se atribuie implicit valoarea 1.

LENGHT(char) Se determină lungimea şirului de caractere **char**.

LOWER(char) Transformă toate literele mari din şirul **char** în litere mici.

LPAD(char1,n[char2]) Cîmpul **char1** este completat la stînga cu o secvenţă de **n** caractere în **char2**; dacă **char2** este omis, se completează cu spaţii.

LTRIM(char,set)... Se returnează cîmpul **char** fără primele caractere, dacă acestea se regăsesc în **set**.

RPAD(char1,n[char2]) Cîmpul **char1** este completat la dreapta cu **n** caractere din **char2**; dacă **char2** este omis completarea se face la dreapta cu **n** spaţii.

RTRIM(char,set) Se returnează cîmpul **char** fără caracterele finale ce se regăsesc în **set**.

SOUNDEX(char) Se selectează cuvintele care se pronunţă asemănător cu şirul **char**.

SUBSTR(char,m[n]) Se face extragerea unui subşir de lungime **n** din şirul **char**, începînd cu poziţia **m** a lui (dacă **n** este omis, extragerea se face pînă la sfîrşitul şirului).

TRANSLATE(char,from,to) Şirul **char** este mutat din şirul de caractere **from** în şirul de caractere **to**.

UPPER(char) Transformă toate literele mici din şirul **char** în litere mari.

USERENV(char) Furnizează informaţii despre utilizator.

Scanned with OKEN Scanner

C. Funcţii de grupare

Funcţii	Valoare returnată
AVG([DISTINCT|ALL]n)	Se calculează media valorilor din n; valorile NULL sînt ignorate.
COUNT([DISTINCT |ALL|]*(expr)	Se numără rîndurile care satisfac expresia expr. Dacă se foloseşte * se numără toate rîndurile.
MAX([DISTINCT| ALL] expr)	Se calculează valoarea maximă din expr.
MINI([DISTINCT| ALL] expr)	Se calculează valoarea minimă din expr.
STDDEV([DISTINCT | ALL]n)	Deviaţia standard din n, ignorînd valorile nule.
SUM([DISTINCT | ALL] n)	Se calculează suma valorilor din n.
VARIANCE ([DISTINCT | ALL|n)	Variaţia din n; valorile nule sînt ignorate.

D. Funcţii de conversie BD

Funcţii	Valoare returnată
CHARTOROWID(char)	Schimbă şirul char în identificator de rînd.
HEXTORAW(char)	Converteşte şirul char, care trebuie să reprezinte un număr hexazecimal, într-o valoare binară.
RAWTOHEX(raw_id)	Converteşte identificatorul de rînd raw_id într-o valoare char, reprezentînd un număr hexazecimal.
ROWIDTOCHAR(row_id)	Converteşte identificatorul de linie row_id într-o valoare caracter.

Scanned with OKEN Scanner

TO_CHAR(expr[,fmt]) Converteşte **expr**, care reprezintă o valoare de tip dată calendaristică, într-un şir de caracter conform formatului **fmt** (vezi Anexa 3 - punctul G). Dacă **fmt** este omis, data este convertită conform formatului implicit (**ZZ/LL-AA**).

TO_DATE(char[,fmt]) Converteşte o dată calendaristică exprimată ca şir de caractere (**char**), într-o valoare numerică **fmt** specifică formatului şirului **char**. Dacă **fmt** este omis, şirul **char** trebuie să aibă formatul implicit de dată calendaristică.

TO_DATE(n,fmt) Converteşte o valoare numerică (**n**) într-o dată calendaristică exprimată sub formă de şir de caractere, cu formatul **fmt**.

TO_NUMBER(char) Converteşte valoarea şirului **char** într-o valoare numerică.

E. Funcţii referitoare la prelucrarea datei calendaristice

Funcţii	Valoare returnată

ADD_MONTHS(d,n) La data **d** se adună algebric **n** luni.

LAST_DAY(d) Se returnează data calendaristică a ultimei zile a lunii precizată în data calendaristică **d**.

MONTHS_BETWEEN(d,e) Se calculează numărul de luni cuprinse între datele calendaristice **d** şi **e**. Dacă **d** este mai mare decît **e**, rezultatul este pozitiv; dacă nu, rezultatul este negativ.

NEW_TIME(d,a,b) Se returnează data calendaristică şi timpul din zona **b**, cînd data şi timpul din zona **a** sînt egale cu **d**. **a** şi **b** sînt şiruri de caracatere ce identifică zone de tip timp.

NEW_DAY(d,char) Se returnează data primei zile a săptămînii

Scanned with OKEN Scanner

numite prin **char**, care este egală sau mai mare cu **d**.

F. Funcţii diverse

Funcţii	Valoare returnată

DECODE (expr,search1,return1,search2,return2,... [default])

Dacă **expr** este egală cu valoarea cîmpului **search1** returnează valoarea **return1**, dacă nu, returnează valoarea **default**.

DUMP(expr[,radix[,start-position[, bytes]]])

Afişează conţinutul zonei de date interne a sistemului ORACLE.

GREATES(expr,expr,....) Returnează cea mai mare valoare dintr-o listă de expresii.

LEAST(expr,expr,...) Returnează cea mai mică valoare dintr-o listă de expresii.

NVL(expr1,expr2) Dacă **expr1** este nulă, returnează **expr2**; dacă **expr1** nu este nulă, returnează **expr1**. **expr1** şi **expr2** pot fi de orice tip.

VSIZE(expr) Returnează numărul de baiţi din reprezentarea internă ORACLE a **expr**.

G. Modele de format pentru TO_DATE si TO_CHAR

Model de format	Valoare returnată
SCC sau CC	Secol;S prefixează data BC cu -
YYYY sau SYYYY	An; S prefixează data BC cu
YYY, YY sau Y	Ultimele 3,2 sau 1 cifre ale anului.
Y,YYY	Anul cu caracterul , pe poziţia punctului zecimal.

Scanned with OKEN Scanner

SYEAR sau YEAR	An, în clar; S prefixează data BC cu -.
BC sau AD	Indicatorul BC/AD
B.C. sau A.D.	Indicatorul BC/AD cu puncte.
Q	Trimestru al anului.
MM	Lună a anului.
MONTH	Numele lunii completat cu blancuri pe o lungime de 9 caractere.
MON	Se returnează abrevierea numelui lunii la trei litere.
WW sau W	Săptămînă a anului sau lunii.
DDD DD sau D	Zi a anului sau a săptămînii.
DAY	Numele zilei, completat cu blancuri pe o lungime de 9 caractere.
DY	Numele zilei (3 litere abreviere).
J	Zi iuliană (zile începînd din 31 decembrie - 4713 BC).
AM sau PM	Indicatorul meridian.
A.M. sau P.M.	Indicatorul meridian cu puncte.
HH sau HH12	Oră a zilei (1-12).
HH24	Oră a zilei (0-24).
MI	Minutul din oră.
SS	Secunda din minut.
SSSSS	Secunde peste miezul nopţii.
/.,etc.	Punctuaţia este reprodusă în rezultat.
"..."	Şirul dintre apostrofuri este reprodus în rezultat.

Formele de prefix de mai jos pot fi adăugate la elementele de format pentru data calendaristică.

fm	"Fill mode". Prefixat la MONTH sau DAY, suprimă blancurile, rezultatul avînd lungime variabilă.

Scanned with OKEN Scanner

Formele de sufix de mai jos pot fi adăugate la elementele de format pentru data calendaristică.

TH	Numeral ordinal (ex."DDTH" pentru "4TH").
SP	Număr explicit (ex."DDSP" pentru "FOUR").
SPTH sau THSP	Numeral ordinal explicit (ex. ,"DDSPTH" pentru "FOURTH"). Formatul implicit este "DD-MON-YY" (ZI-LUNĂ-AN).

Un element ce specifică cuvintele explicite poate fi exprimat cu litere mici sau mari pentru ca SQL să afişeze rezultatul în acelaşi fel.

H. Pseudo-coloane

Număr coloană	Valoare returnată
LEVEL	1 pentru nodul rădăcină, 2 pentru un subdirector (copil) al rădăcinii etc. Se utilizează în comanda SELECT şi clauza CONECT BY.
NULL	Returnează o valoare nulă. Nu poate fi folosită în expresiile logice.
ROWID	Numărul de identificare al rîndului. Un identificator de rînd este de tipul ROWID şi nu de tipul număr sau caracter.
ROWNUM	Numărul de ordine a rîndului selectat din tabelă (numărînd de la 1).
SYSDATE	Data calendaristică şi timpul curent.
UID	Identificator utilizator; este un număr unic pentru fiecare utilizator.
USER	Returnează numele utilizatorului curent.

Scanned with OKEN Scanner

ANEXA 4
DICŢIONARUL DE CUVINTE/EXPRESII

Dicţionarul de cuvinte conţine informaţii despre definiţiile din baza de date. Utilizatorul are acces la aceste informaţii printr-o serie de consultări ale dicţionarului, folosind următoarele nume de viziuni în comanda SELECT:

Nume de viziune	Descriere
AUDIT_ACCES	Ieşirea AUDIT a sistemului ORACLE stabileşte legatura cu tabela de acces curentă sau cea mai recentă a utilizatorului. Pentru un DBA ieşirea AUDIT a ORACLE stabileşte legătura cu toate tabelele de acces, curente sau cele mai recente, ale utilizatorilor.
AUDIT_ACTIONS	Codurile acţiunilor AUDIT şi descrierea lor.
AUDIT_CONNECT	Ieşirea AUDIT a ORACLE stabileşte legătura cu sesiunea curentă sau cea mai recentă a utilizatorului. Pentru un DBA ieşirea AUDIT a ORACLE stabileşte legătura cu sesiunea curentă sau cea mai recentă a utilizatorului.
AUDIT_DBA	Ieşirea AUDIT a ORACLE stabileşte legatura numai cu operaţiile DBA.
AUDIT_EXISTS	Ieşirea AUDIT a ORACLE stabileşte legatura cu tabela accesată care eşuează cu *"nu există"*.
AUDIT_TRAIL	Ieşirea AUDIT a ORACLE stabileste legătura cu utilizatorul. Pentru un DBA, ieşirea AUDIT a ORACLE stabileşte legatura cu toţi utilizatorii.
CATALOG	Tabele, viziuni şi clustere accesibile utilizatorului, mai puţin însuşi dicţionarul bazei de date.
CLUSTERS	Clusterele utilizatorilor.
CLUSTERCOLUMNS	Coloane în clusterele utilizatorilor.

Scanned with OKEN Scanner

COL Coloane în tabelele utilzatorilor.

COLUMNS Coloane în tabele accesibile utilizatorilor.

DBLINKS Legaturi ale bazei de date accesibile utilizatorilor.

DEFAULT_AUDIT Opţiuni ale AUDIT-lui ORACLE pentru noile tabele create. Pentru un DBA, opţiunile AUDIT-ului ORACLE stabilesc legături cu toate tabele utilizatorilor.

DTAB Tabele şi viziuni în dicţionarul de date.

EXTENTS Structura de date a extensiilor din tabele.

INDEXES Indecşi creaţi de utilizator.

PARTITIONS Structura fişierelor din partiţie. Are acces numai DBA.

PRIVATESYN Sinonimele utilizatorului.

PUBLICSYN Sinonimele publice.

SESSIONS Indică utilizatorii conectaţi la o bază de date.

SPACES Defineşte spaţiul pentru crearea de tabele şi clustere.

STORAGE Alocarea datelor şi indecşilor în memorie pentru tabelele utilizatorilor.

SYNONYMS Sinonimele utilizator şi publice.

SYSCATALOG Tabele, viziuni şi clustere accesibile utilizatorilor.

SYSCOLUMNS Coloane în tabele şi viziuni accesibile utilizatorilor.

SYSEXTENTS Structura datelor tabelei din sistem.

SYSINDEXES Indecşii creaţi.

SYSPROGS Lista modulelelor de acces ale utilizatorilor (cele care nu sînt utilizate curent).

SYSSESIONS Arată toţi utilizatorii conectaţi prin *logon*.

SYSSTORAGE Rezumat al memoriei bazei de date (numai pentru DBA)

SYSTABALLOG Spaţiul de alocare a tabelelor pentru date şi indecşi.

Scanned with OKEN Scanner

SYSTABAUTH	Informaţii privind accesul autorizat la baza de date.
SYSTEM_AUDIT	Informaţii despre resursele de utilizare ale sistemului pe care ORACLE le returnează la comanda AUDIT.
SYSUSERAUTH	Lista principală a utilizatorilor ORACLE (numai de DBA).
TAB	Tabele, viziuni şi clustere aparţinînd utilizatorului.
TABALLOC	Alocarea datelor şi indecşilor din memorie pentru tabelele utilizator.
TABLE_AUDIT	Opţiuni AUDIT referitoare la utilizator. Pentru DBA, opţiunile AUDIT ale ORACLE se referă la toţi utilizatorii.
VIEWS	Text de cerere într-o comandă CREATE VIEW care creează fiecare viziune în baza de date.

Scanned with OKEN Scanner

BIBLIOGRAFIE

Alfonso F.Cardanes, *Data Base Management Systems*, Allyn and Bacon.Inc., Boston, London, Sydney, Toronto, 1985.

C.J.Date, *An Introduction to Data Base Systems*, Edition 4, Addison Wesley Publishing Company, 1986, London, Amsterdam, Sydney.

H. Loomis, *Data Management and File Structures*. The Benjamin Cummings Publishing Company, California, 1989.

Gh.Sabău, V.Avram, I.Lungu, *Sisteme de gestiune a bazelor de date*, Academia de Studii Economice, Bucureşti, 1983.

Gh.Sabău, I.Lungu (coordonatori), *Sisteme informatice şi bănci da date*, Academia de Studii Economice, Bucureşti, 1992.

*** ***ORACLE Application Tools for MS-DOS. User's Guide.*** Oracle Corporation, 1988, U.S.A.

*** ***ORACLE Application Tools for MS-DOS. Getting Started.*** Oracle Corporation, 1988, U.S.A.

*** ***SQL Language Reference Manual***, Oracle Corporation, 1988, U.S.A.

*** ***SQL*PLUS Version 2.1. User's Guide Manual***, Oracle Corporation, 1988, U.S.A.

*** ***SQL*PLUS Version 2.1. Reference Manual***, Oracle Corporation, 1988, U.S.A.

*** ***SQL*PLUS Version 2.1. Reference Card***, Oracle Corporation, 1988, U.S.A.

Scanned with OKEN Scanner

COMPUTERELE LUCREAZĂ!.
CÎŞTIGUL DVS. ESTE CERT!

BURSA DE INFORMAŢII COMERCIALE "ALL"

Aici calculatorul vă răspunde prompt dacă îl solicitaţi!

Furnizori, beneficiari, intermediari, parteneri de afaceri din toată ţara, din orice domeniu pot fi imediat la dispoziţia dumneavoastră.

ABONAŢI-VĂ AZI!

Vă oferim GRATUIT serviciile noastre timp de o lună !

Cu taxe simbolice veţi găsi, numai prin noi

CONEXIUNI EXCELENTE

Informaţii şi abonamente la **BIC ALL SRL**
Bucureşti, Calea Victoriei nr.120, CP 12-107
Tel: 90/50.44.20, Fax: 90/12.34.07

Scanned with OKEN Scanner

În Editura ALL au apărut:

Biblioteca de economie şi management

Asigurări şi reasigurări în comerţul internaţional - *Constantin Alexa, Violeta Ciurel*

Doctrine şi curente în gîndirea economică modernă şi contemporană - *Sută Sultana Sălăjan*

Dicţionar de sociologie - *Lizetta Coandă*

Biblioteca juridică

Codul civil - ed. îngrijită de *Corneliu Bîrsan, Valeriu Stoica, Flavius Baias*

Drept civil. Teoria generală a obligaţiilor - *Constantin Stătescu, Corneliu Bîrsan*

Biblioteca medicală

Farmacologie - sub redacţia *Valentin Stroescu*

Anatomia omului - vol. I - Aparatul locomotor - *Victor Papilian*, Ediţia a VII-a -revizuită şi adăugită de *Ioan Albu.*

Parazitologie medicală - *Simona Rădulescu, Ernest A. Mayer*

Dialog social

Sistemele axiomatice ale logicii limbajului natural. Funcţii şi operaţionalizare - *Gabriel Andreescu*

Biblioteca de arhitectură

Fiinţa şi spaţiul - *Radu Drăgan şi Augustin Ioan*

Scanned with OKEN Scanner

Editura **ALL** va tipări:

Biblioteca de informatică

COBOL COMPLET - Vol. 1 - Bazele programării în COBOL pentru calculatoare IBM - PC; Vol. 2 - Tehnica programării în COBOL pentru calculatoarele IBM - PC - *Ion Gh. Roşca şi col.*

Biblioteca de economie şi management

Comportamentul uman. O abordare economică - *Gary S. Backer*

Biblioteca medicală

Clinică medicală. Analize şi sinteze Vol. I şi II - *Simion Purice*

Stresul psihic şi bolile interne - *Ioan Bradu Diamandescu*

Îndreptar pentru prescrierea medicamentelor - *sub red. V. Stroescu*

Explorarea ultrasonică a sistemului vascular (neurologie) - *Andrei Nistorescu*

Anatomia omului Vol. II - Splanhologia - *Victor Papilian, ed. a VI-a* - revizuită şi adăugită de *Ion Albu*

Biblioteca juridică

Tratat de dreptul familiei - *Ion P. Filipescu*

Dialog social

Introducere în convenţia europeană a drepturilor omului - *Donna Gomien*

Metafizica lumilor posibile şi existenţa lui Dumnezeu - *Adrian Miroiu*

Biblioteca de matematică

Lecţii de analiză matematică - *Paul Flondor, Octavian Stănăşilă*

Matematici speciale - *Octavian Stănăşilă, Vasile Brînzănescu*

Analiză matematică - culegere de probleme Vol. I şi II - *D. Flondor, N. Donciu*

Scanned with OKEN Scanner